DESIGN and ANALYSIS OF ROUTER ARCHITECTURES FOR NoC

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

VLSI DESIGN AND EMBEDDED SYSTEM

BY

Siddhardha Pottepalem

213EC2208

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

ODISHA, INDIA. 769008

2015

DESIGN and ANALYSIS OF ROUTER ARCHITECTURES FOR NoC

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

VLSI DESIGN AND EMBEDDED SYSTEM

BY

Siddhardha pottepalem

213EC2208

Under the Guidance of

PROF. Ayas Kanta Swain

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

ODISHA, INDIA. 769008

2013 – 15

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA
ODISHA, INDIA- 769008

CERTIFICATE

This is to certify that the thesis report entitled

DESIGN and ANALYSIS OF ROUTER ARCHITECTURES FOR NoC

submitted by

Mr. Siddhardha Pottepalem

bearing roll no. **213EC2208**
in partial fulfilment of the requirements for the award of

MASTER OF TECHNOLOGY in
VLSI DESIGN AND EMBEDDED SYSTEM

during session 2013-15 at National Institute of Technology, Rourkela, is an authentic work carried out by him under my supervision and guidance.
To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university/institute for the award of any degree or diploma.

Prof. Ayas kanta Swain
Department of ECE,
National Institute of Technology,
Rourkela-769008.

Date: June 2nd 2015

Dedicated to
my best two years at NIT Rourkela.

Abstract

The advance of process technology keeps on reducing the device size. As a result, the number of processing elements that can be integrated on a single chip (SoC) increases. The reduction in the device size also reducing the gate delay compared to the wire delay giving rise to increase in the frequency of operation of the devices. Further, in order to reduce the design time to market the communication system must support the plug and play architecture and should support design reuse. The conventional on-chip communication architecture, which consists of point-to-point connection and bus infrastructure, may not be able to provide sufficient communication requirements for SoC in terms of increasing the frequency of operation, providing reliability and flexibility. Further, conventional communication systems used for on-chip communication are not scalable and does not support design reuse.

The NOC design represents a new paradigm to design multi-processor SoC which is scalable and supports design reuse. The NOC architecture uses layered protocols and packet switched networks which consist of on-chip routers, links and network interface on a predefined topology. NoC requires many on-chip resources which can increase the cost, area and power consumption. The efficiency of the NoC depends on how the resources are utilized for traversing the packet from source to destination which is determined by the flow control mechanism. The components which are used in the NoC for establishing communication between the modules of SoC were designed using VERILOG HDL. Different types of router architectures used by NoC were also designed mentioning their merits and demerits and their area and power consumption was also estimated.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my project guide, **Prof. Ayas Kanta Swain,** for his excellent guidance, patience and providing me with an excellent atmosphere for doing project throughout the course. The regular discussions with him and his ideas for solving the problems I faced helped a lot in this project. Without his help this dissertation would not have been possible.

Next, I would like to thank **Prof. K. K. Mahapatra**, **Prof. D. P. Acharya**, **Prof. P. K. Tiwari**, and **Prof. M. N. Islam** for their valuable advices and ideas which helped me to think clearly whenever I got stuck during my project.

I would like to thank all the faculty members and non-teaching staff of the Department of ECE for helping me directly or indirectly in completion of this project.

Next, I would like to express my thanks to all the Ph. D. and Research scholars who have readily provided a helping hand in my time of need and acknowledged my numerous requests regarding any technical support.

The most important people of my life are my parents. They have been a source of inspiration throughout my life. They guided me and helped me during my toughest situations in my life and sacrificed a lot for me. I would like to thank them from the bottom of my heart.

Away from home in a distant land, the people who constantly support and encourage a person are his friends. I am utmost grateful to my friends, here at NIT Rourkela. Apart from the regular education from my professors, I have learnt many things from my friends. I would like to express my thanks to all of them for making these two years a fantabulous journey.

SIDDHARDHA

Contents

List of figures

List of Tables

Abbreviations

ASIC	Application Specific Integrated Circuit
BA	Bus Arbiter
CLB	Combinational Logic Block
FIFO	First-In-First-Out
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HoL	Head of Line
NI	Network Interface
NoC	Network-on-Chip
PE	Processing element
RC	Routing Computation unit
SA	Switch Arbiter
SoC	System-on-Chip
VC	Virtual Channel
VCA	Virtual Channel Allocator
VCI	Virtual Channel Identifier
RTL	Register transfer logic

1. Introduction

With the scaling of the technology, multi-core processors are able to integrate on a single chip termed as system on chip (SoC). Previously conventional interconnects like point-to-point interconnects and buses were used to establish the communication between different processing elements (PEs) of a chip. The scaling of technology also lead to the decrease in the gate delays giving rise to increase in the frequency of operation of the processors resulted in high speed operation [1]. Further, pipeline architectures for the processing of information caused further reduce in latency. The communication architecture must also support the design reuse in case of application specific integrated circuits (ASIC) to reduce the design time to market.

1.1. Conventional communication system

It includes Point-to-point interconnection, Crossbar connection and Bus connection.

1.1.1. Point-to-Point connection

In this type of connection there is an independent path from one node to another that is, a node is connected with all the remaining nodes in a direct path as shown in figure 1.1.

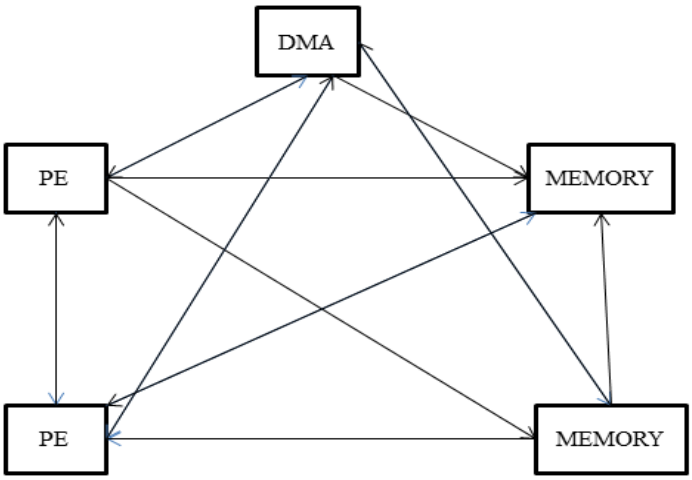

Figure 1.1: point-to-point connection

The performance of this system in establishing the communication is high. Whenever there is data available at a node data can be transferred immediately without any requirement to wait, as there is a dedicated path between every source node and destination, thus offering low latency. The disadvantage of this is even though the data is not communicated between the nodes a path will be there between the nodes which is not desired [21]. Further an *n* node network requires $\frac{n(n-1)}{2}$ connections between the nodes giving rise to consuming more area and power as the value of *n* increases.

1.1.2. Crossbar connection

In this connection each element is connected to the other by a crossbar network. The communication established between two elements lying at opposite stages of the crossbar network is done by establishing a connection at a cross point. This cross point connection is established by exciting the corresponding row and corresponding column. The view of the crossbar connection is as shown in figure 1.2.

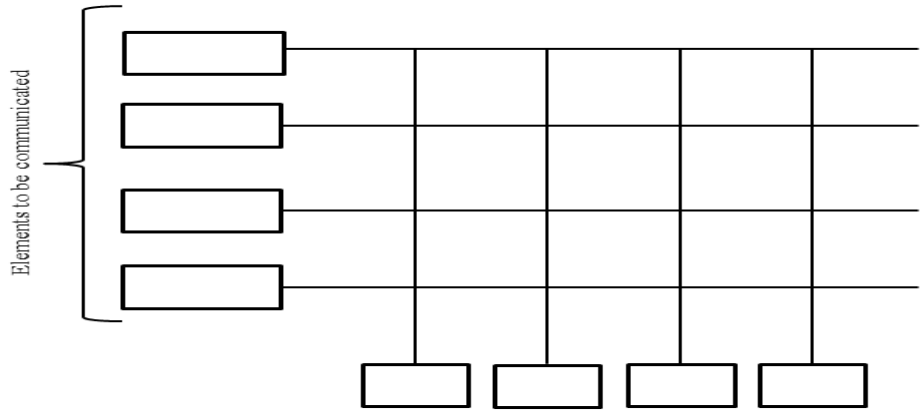

Figure 1.2: crossbar connection

The drawback of this is it is not possible to establish the communication between the components of the same stage. Further it causes output blocking which results loss of the data. To avoid the loss of data due to output blocking a buffer is needed to be placed at each cross point.

1.1.3. Buses

In this only one source node and destination node can communicate with each other at a time. Before transmission of data all the data transferring elements must check whether the bus is idle or busy. If the bus is idle node elements can transfer the data, when bus is busy the node elements must wait until the bus becomes idle. When the bus is idle sometimes the node elements tries to access the bus simultaneously giving rise to conflicts. These conflicts can be resolved by using an arbiter. The bus communication system is as shown in the figure 1.3.

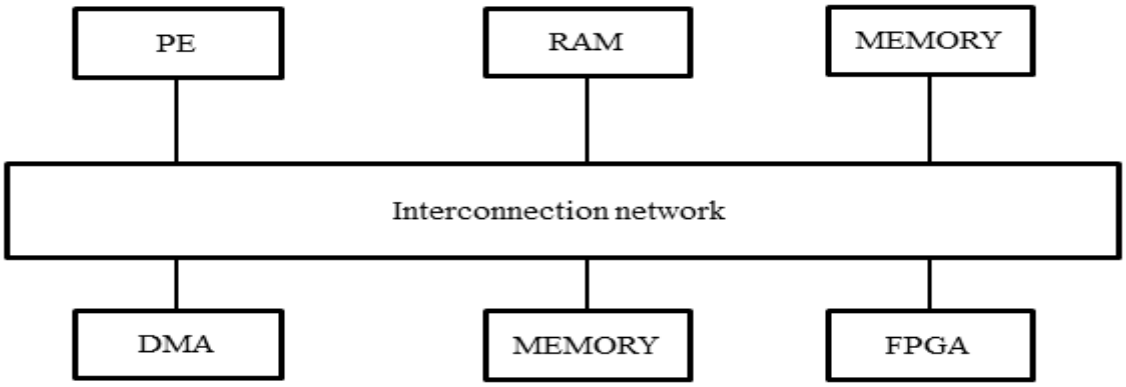

Figure 1.3 bus communication system

1.2. Motivation

The basic building blocks of any digital system consist of logic, memory and communication. Logic manipulates data, memory stores the data for feature use and communication transfers data between different modules of the digital system. The communication system decides the performance, reliability and speed of the digital system. Thus a good communication system is necessary to have a high performance digital system with good quality of service (QOS) which describes throughput, latency and dropping probability.

In the present technology era, gate delays are becoming very much less than the wire delays. Moreover the conventional communication systems are not scalable, does not support high frequency of operation and design reuse. To overcome these disadvantages a new paradigm network on chip (NoC) was proposed. It uses packet based communication operation offering design reuse, scalability, flexibility and good QOS. The advantages offering by the NoC has made this as an active area in research over a decade. NoC is used to establish communication between different PEs of SOC.

NoC uses large number of resources consisting of buffers, network interface (NI) and routers which consists of many logic components. This increases the cost and area of NoC. The effective utilization of the resources used by the NoC in way that it can reduce the cost, area and power consumption is the major research area in the NoC.

1.3. Objective

- To understand various concepts of NoC including the drawbacks of the NoC design and to go through the design parameters of the NoC.
- To design a NI that can effectively transfer data between different clock cycles and between different processors having different data widths.
- To design a router which is free from starvation, deadlock and head of line blocking (HoL) by reducing its area and power.

1.4. Organization of the thesis

The theme of this thesis is to discuss the designing of the components of NoC and different NoC router architectures and to evaluate its area and power consumption.

Chapter 1 discusses the conventional communication system and there drawbacks and motivation in choosing the work on NoC.

Chapter 2 presents the overview of NoC. It describes about different switching techniques, topologies and includes literature survey.

Chapter 3 discusses the necessity of generic network interface, using the wishbone compatible asynchronous FIFO for the design of NI and metastability problem in digital system.

Chapter 4 presents different router architectures, the components required for the design of the router architecture and discusses the merits and drawbacks of buffered and bufferless router.

Chapter 5 shows the simulated output waveforms, device utilization summary, area and power consumption of the different components of NoC.

2. Overview of NoC

Following Moore's law the technology is scaling down this resulted in increasing in the density of the chip which makes it possible to accommodate large number of processors onto a single chip termed as system on chip (SOC).

The basic building blocks of any digital system are memory logic and communication. Memory is used to store the data that should be processed. Logic performs some operations on the data and manipulates the data to get the desired result. Communication transfers data from one location to other location.

The performance of any digital system mainly depends on communication which determines speed, latency, throughput and quality of service (QOS) [2]. Hence, to have a good system performance an effective communication is required.

As the number of processing elements and the frequency of operation in an SOC increases conventional busses are not able to establish the communication to achieve the desired performance. Further, busses are not scalable; this motivates to introduce a new communication paradigm network on chip (NoC).

NoC uses packet based communication architecture. Using graph theory NoC can be viewed as graph consisting of nodes and edges. Each node of an NoC consists of processing element (PE), network interface (NI) and router. Edges correspond to the communication links (channels) composed of wires and these channels connect the routers for transferring the data between them. A NoC link between two routers consists of two physical channels to support the full duplex communication.

2.1. Design specifications of NoC

The design specifications of NoC include

1. Switching techniques

2. Topology
3. Routing algorithm
4. Flow control

2.2. Switching techniques

Communication network generally have multiple path between the source and destination to transfer the packets. Devices of the network are designed to carry out the function of allocating a particular path out of the possible multiple alternative paths. This path establishment based on a particular call is done by switching; NoC is a switched network in which a switch connects atleast two links together. There are two different types of switching

1. Circuit switching
2. Packet switching

2.2.1. Circuit switching

In this a dedicated connection is always established between the two end systems even if the data is not transferred. A switch can make this dedicated connection (circuit) active or inactive. The advantage of this is it offers minimum latency, packets are delivered in order and the number of retransmissions are very less. The disadvantage of this is for an *n* node network it requires ${}^{n}C_{2}$ channels. Due to dedicated connection the resources are underutilized. As the number of nodes increases area and power consumption of the interconnect increases.

2.2.2. Packet switching

In this the communication between the two ends is done in blocks of data called packets instead of continuous communication. In this a path is shared by many users. There are two modes of packet switching. In the first mode each and every packet carries source and destination addresses and forwarded by using some algorithm. This process is continues until all the packets reach the destination. In the second mode a path is established first and all the packets for that particular call follows the same path one after another whenever the forward links are available.

A connection is established only when source wants to communicate with the destination. The connection establishment can be connectionless or connection-oriented. In case of connectionless communication establishment a connection is not required between source and destination. Packets are routed into different routes to reach the destination in this packet ordering is required. In case of connection-oriented mechanism a connection is established temporarily between source and destination for the packet transmission and the connection is released when the packet transmission is completed. Packet switching techniques include store-and-forward, virtual cut through and wormhole switching [4].

- **Store and forward switching**

A network node must receive the entire packet before forwarding it to the other node. The disadvantage of this is due to waiting for the entire packet to receive delay increases, a large amount of buffer size is required to store the entire packet on the node. If a portion of packet is dropped the entire packet must be retransmitted; which is undesired.

- **Virtual cut through switching**

Unlike store and forward it does not wait for the entire packet to be stored. When a network node receives a portion of packet it forwards to the downstream node, if the buffer space is available in the next node. Hence the delay is reduced but still the minimum size of the buffer should be able to store an entire packet.

- **Wormhole switching**

A packet is further divided into flits known as flow control digits. A flit is the smallest unit on which flow control is performed. Flits are delivered in a pipeline fashion through the network. Same as virtual cut through switching the entire packet is not stored. Hence the size of buffers to store the flits very less which reduces a huge amount of area and power which is favoured for NoC. Further, latency is very less as there is no need for waiting for the entire packet. Wormhole switching employs connection-oriented scheme so, there will be no need of packet ordering. In this flow control is done at flit level and switching is done at packet level.

The key to the efficiency of interconnection networks comes from the fact how resources are shared. Flow control decides which messages get access to particular network resources overtime mainly during contention. A good flow control should forward packets with minimum delay and avoid idling of resources during heavy loads.

2.3. Topology

It refers to the way network nodes are connected i.e, NoC consists of a collection of routers and channels; the way in which these channels and routers are connected are referred to as topology. NoC can be assumed as a network of roadways, road lines indicates channels, the vehicles travelling through the road ways can be assumed as packets and the

interconnection of the roadways as the router. The way how the roads are connected refers to topology. NoC topologies are categorized as direct topology and indirect topology

2.3.1. Direct network topology

In direct network topology nod processors are directly connected with each other by the network, each node performs routing as well as arbitration. Direct topology is also known as orthogonal topology. The number of switches (routers) and number of resources (ex. PEs, memory) are same since only one resource is connected to a switch. It affects routing, reliability, throughput, latency and ease of building the network.

- **Mesh topology**

These are highly regular hence the interconnect length between nodes will be uniform to ensure performance uniformity. This regular physical arrangement is well matched to packaging constraints. In Mesh apart from the corners each node which consists of processing elements, routers and network interface is linked to four other nodes by communication channels.

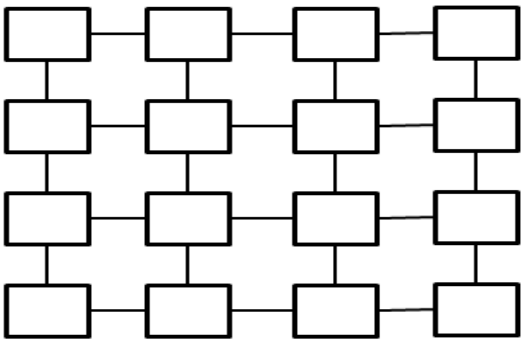

Figure 2.1: 4X4 Mesh topology

- **Torus topology**

It is similar to the mesh topology except the routers at the boundary are connected differently. These are wrapped around from top routers to bottom routers and rightmost to leftmost resulting in doubling of bandwidth as shown in figure 2.2 thus in Torus topology each node is connected to four neighbours this increases path diversity. It is harder to layout on chip and possesses unequal link lengths.

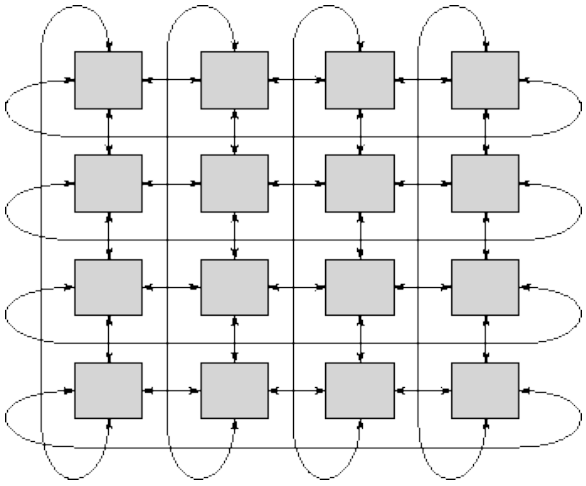

Figure 2.2: 4X4 Torus topology

2.3.2. Indirect network topology

In this node processors are connected by one or more intermediate node switches, the switching node performs the routing and arbitration. There will be no direct connection between processors.

- **Butterfly topology**

A butterfly topology has a minimum diameter. The drawbacks of butterfly topology are

It has no path diversity i.e, there is exactly one route from each source to each destination node. It cannot be realized without long wires. These long wires make butterflies less attractive to moderate sized large interconnection networks.

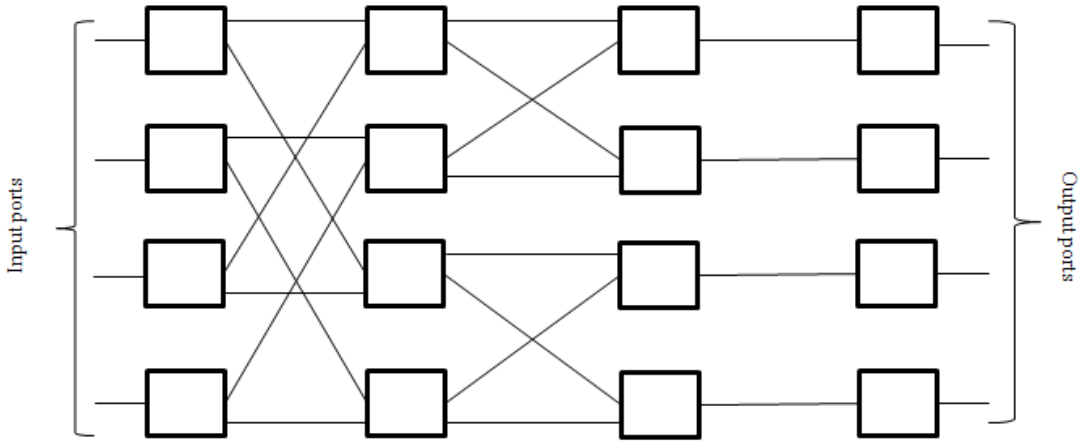

Figure 2.3: 4X4 Butterfly topology

2.4. Routing algorithm

Once we have topology of our network the next step is routing. Selection of a single path out of many alternative paths and switching of the packets is routing it can be static or dynamic. A good routing algorithm should balance load across the network even if the traffic is non-uniform. The more balanced the load, the closer the throughput of the network to be idle. A well designed routing algorithm must also keep path lengths as small as possible. It should reduce the hop count and overall latency of the message.

2.4.1. Taxonomy of routing algorithms

Routing algorithms are classified based on how they select a path between set of possible paths R_{xy} from source x to destination nod y.

- **Deterministic routing algorithm**

It always chooses the same path from source x to destination y even if there are multiple paths to travel. Implementation of deterministic routing algorithms is very simple and easy to make deadlock-free. The disadvantage is that these are not able to balance the load across the links in non-uniform or busty traffic and they ignore path diversity. This algorithm is the best choice for uniform or regular traffic patterns.

- **Source routing**

In this scheme the entire route of the packet is decided by the source router. As the packet traverses in the network this information is used by each router on the path to forward the packet towards the destination. The problem with this is it requires large overhead which reduces the bandwidth utilization of the network.

- **Distributed routing**

The routing decision is taken at individual routers depending on different parameters while the header of the packet includes only destination address. The complexity of implementation of distributed routing is more.

- **Oblivious algorithms**

These routing algorithms include deterministic routing algorithms as a subset. It chooses a path without any information about the present state of the network. It is simple to implement

- **Adaptive algorithms**

These algorithms adapt to the state of the network. The route of the packet from source x to the destination y is determined by the network conditions. Thus it can decrease the probability of passing a packet from a congested or malfunctioning link. This algorithm is preferable in presence of irregular traffic or in networks with unreliable routers and links. The disadvantage of this is it could not guarantee the order of packets, for this it requires a reordering module. This increases design complexity and latency.

2.5. Flow control

It determines how the network resources such as buffers and channel bandwidths are allocated to the packets which are traversing through the network. A good flow control should allocate these resources to achieve high throughput and low latency. Flow control can be viewed as a technique of resource allocation or resolving contention. From resource allocation point of view it should allocate channel bandwidth and buffers for the forwarding packets. This same process can be viewed as contention resolution. For example when two packets tries to access the same resource simultaneously the flow control technique resolves this problem by allocating the channel to one packet and blocking the other packet. The blocked packet is stored in a buffer in case of buffered flow control and misrouted or dropped in case of bufferless flow control.

2.6. Literature survey

To overcome the drawbacks of bus based interconnect systems in SoC systems, researchers are looking for a substitute for this that can increase the performance of the SoC systems.

The introduction to NoC and the drawbacks of the conventional communication systems are explained in [1]. The architecture and design of the wormhole router were explained by the author of thesis in [23].

To overcome the drawback of wormhole router including HoL blocking and deadlock a virtual channel router was proposed in [11]. It fails in effectively utilizing the resources due to statistically allocated buffer. Instead of using statistically dedicated buffers to the virtual channels dynamic virtual allocator which reduces number of VCs and increases the depth of FIFO during low traffic and increases the number of VCs and reduces the depth of FIFO during heavy traffic was proposed by the authors in [16].

Dynamically allocated multi queue buffers using linked lists were proposed by the authors of paper [14]. The drawback of this is it uses three clock cycles to forward packets and the design complexity is very high. The concept of buffer sharing was introduced in [10], [20] to effectively utilize the on-chip resources.

When contention occurs packets are blocked in the network which in turn gives rise to congestion [22]. An efficient congestion management by building congestion was presented by the authors in [22].

3. Design of NI module

NI provides services to the transport layer of ISO-OSI reference model [9]. It connects the PE to the router. It acts just like the interface card used in computers to connect to the internet. As NoC uses packet based communication the messages that are generated by PEs must be converted into packets before the messages are transmitted into the network to the requested processors or memory elements. This conversion of the messages generated by the processing elements into packets that consist of header, body and tail by adding some required fields which is required for the efficient transfer of the packets from source to destination is done by the network interface. The data is converted in a form that it can traverse through interconnect which are routed by routers. To reduce the design time, to support plug and play architecture and design reuse a protocol based generic NI is required. The key ingredient in plug and play architecture is the decoupling of computation from which requires the interfaces that connect the intellectual property (IP) cores [9]

3.1. Requirements of NI to be generic

- It should support the PE element of any data width, as NoC can have different PEs of different data width using different protocols the NoC should support the reuse of PEs. For this Wishbone bus protocol is chosen as it is open source.
- It should be able to transfer data between different clock frequencies without loss for this asynchronous FIFO is used.

A wishbone compatible FIFO was designed to obtain a generic NI. The maximum data width of the processing element can be 64 bit. As we have used a 64-bit register with 3-bit select line. It can receive and send data of data width ranging from 8 to 64 bit in multiples of 2. In paper [ ] a wishbone compatible generic interface is designed but making FIFO compatible to Wishbone offers more flexibility and less logic circuitry than making NI module compatible to Wishbone.

3.2. Asynchronous FIFO design

FIFO is used as a buffer to store data in a router when contention occurs. This FIFO must support simultaneous read and write operations to reduce latency and it should effectively transfer data from one clock domain to another clock domain [7]. To satisfy all these requirements we have chosen asynchronous FIFO which can effectively transfer data from one clock domain to another clock domain and it supports simultaneous read and write operations.

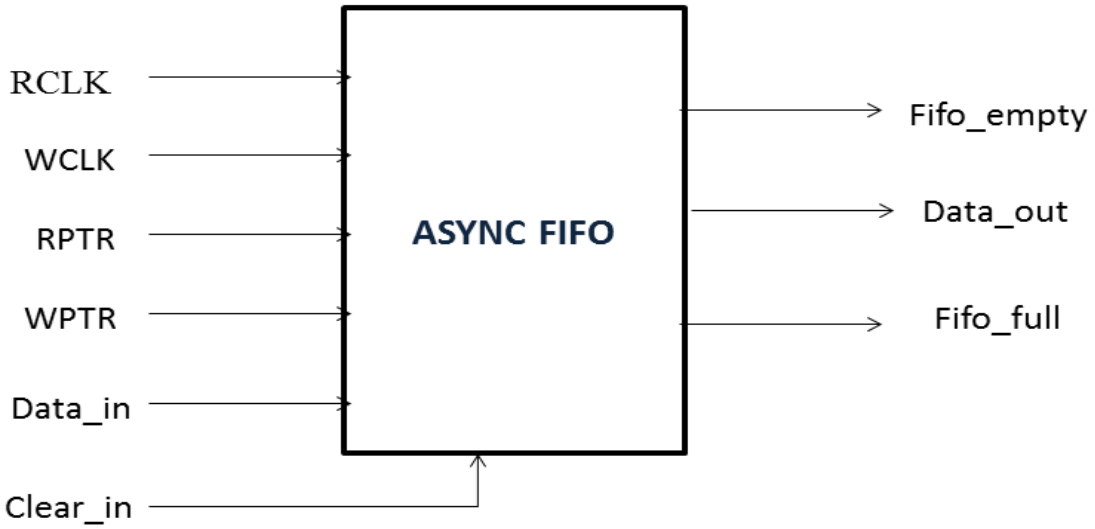

Figure 3.1: input and output signals for asynchronous FIFO

3.2.1. Metastability

It is a phenomenon that causes system failure of digital devices, when a signal is transferred between the circuitry in asynchronous clock domains. All registers in digital circuits have defined timing requirements that allow each register to capture data correctly at the inputs and produce the outputs [7]. For having a reliable operation data should come earlier before the rising edge of the clock which is known as hold time (T_h) and it should not change for certain amount of time after the rising edge of the clock termed as setup time (T_{su})

and the output is produced after certain amount of time which is propagation delay (T_p) of the register.

If the data transition violates the setup or hold time requirements, the register output may go into a metastable state. The output of the register remains between the high and low values for certain time, hence the output transition high or low state is delayed beyond the propagation delay of the register. In synchronous systems the inputs meets the timing specifications of the registers, so metastability problem does not occur in them. It commonly occurs in asynchronous systems. If the data output comes to a valid state before the next register captures data, then metastability does not affect the system. But if the metastable state is not resolved between the output low or high state before it reaches the next register it causes the total system failure.

3.2.2. Asynchronous FIFO pointers

For supporting simultaneous read and write operations FIFO requires read and write pointers. The write pointer always points to the next location to be written. On reset, both the pointers point to zero location which is also next location to be written. FIFO write operation includes two steps, first the memory location pointed by write pointer is written and then the write pointer is incremented to point to the next memory location to be written.

The read pointer always points to the current location to be read from the FIFO. On reset both pointers are set to zero, the FIFO is empty and the read pointer points to the invalid data to be read. As FIFO is empty there will be no data to read. When first data is written into FIFO, the write pointer increments and the empty flag is cleared. The read pointer still pointing to the first FIFO word, it immediately drives the data into FIFO output port.

For incrementing the FIFO pointer a counter is used. Using a binary counter for pointers will not be a good idea because of its disadvantages. Synchronizing the binary count

from one clock domain to the other will be problematic due to the change of every bit simultaneously in a n-bit counter [7]. For example change of value from 7 to 8 that is, 0111 to 1000 all the four bits are changed simultaneously in a single clock transition. This can cause metastability problems since the setup and hold time are not able to meet the timing constraints due to simultaneous bit changes in a single clock pulse.

To rectify this effect simultaneous bit transitions should not take place in a counter. Gray counter suits the best for using as a pointer than binary counter because in gray code only one bit is altered between two words. The gray code counter must have counting values of power of 2. So an n-bit counter can count upto 2^n memory locations or clock pulses. Considering the previous example when the sequence changes from 7 to 8 that is, 0100 to 1100 (gray code of 7 and 8) in which one bit is altered.

3.2.3. Asynchronous FIFO flags

FIFO consists of two flags to indicate the status of the FIFO weather it is full or empty. FIFO empty indicates it does not have any valid data to be read from it and FIFO full indicates any further data write operation will overwrites the data which is not read from the FIFO.

FIFO is empty when both read and write pointers are equal. This happens on reset operation or whenever read pointer catches the write pointer. The FIFO is full again when both the pointers are equal, when the write pointer wraps around and meets the read pointer. The problem in designing of this is which one belongs to empty and which one belongs to full.

To distinguish between full and empty signals an extra bit is added to each pointer. This extra bit which is the pointer MSB is used to differentiate between full and empty signals. When the write pointer increments to the final FIFO address, the write pointer sets

the unused MSB and the rest of the bits to zero as shown in figure 3.2. For read pointer also the same thing is done. If MSB's of both pointers are different, which means the write pointer has wrapped one more time than the read pointer. If the MSB's of both pointers are same, it indicates both the pointers have wrapped same number of times.

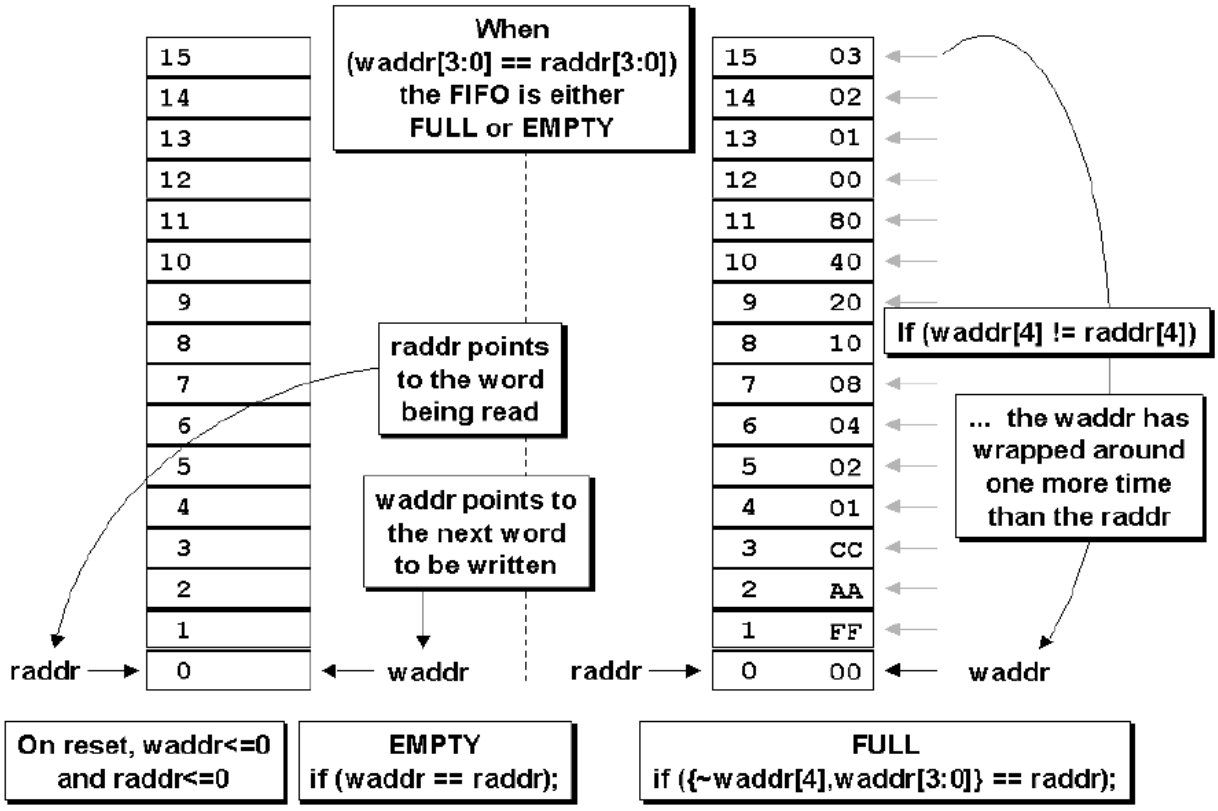

Figure 3.2: generation of FIFO full and empty signals

Thus by using n-bit pointers we can address 2^{n-1} writable and readable memory locations which corresponds to the depth of the FIFO. The FIFO is empty when both pointers including MSB's are equal and full both pointers except the MSB's are equal.

3.3. Making asynchronous FIFO compatible to WISHBONE

Wishbone bus works on the principle of handshaking protocol [6]. Here master is the Wishbone compatible processing element and slave is asynchronous FIFO. To make FIFO compatible with Wishbone bus rst_i,stb_i sel_i and ack_o signals are added to the FIFO. Wishbone master and slave can be connected in three different types of communication modes. In this design wishbone master and slave are connected in point-to-point communication mode as shown in figure 3.3.

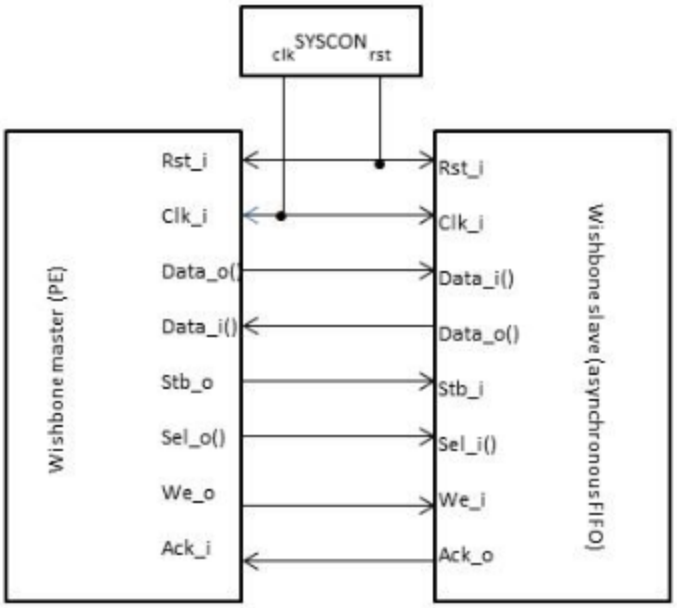

Figure 3.3: Making FIFO compatible to WISHBONE

.

Wishbone bus consists of two modules SYSCON and INTERCON. SYSCON module generates clk_i and rst_i signals for the wishbone master and slave. INTERCON module contains all the connections between master and slave. We used a 3-bit select signal. It helps to determine the data width of the Wishbone master. A 64-bit register is used to latch the data from master upon the raising edge of the strobe (stb_i) signal for validation of the worst case scenario of 64-bit master. This latched data is stored into the asynchronous FIFO by

introducing some wait states during the burst transfer of the data. Slave can put any number of wait states by lowering the ack_o signal until it stores the message received from the latch or when the FIFO is full.

3.4. NoC packet format

We have used wormhole switching for the design. The messages are converted into packets and the packets are still divided into flits which are the flow control digits. Flits are the smallest flow control digits that traverse through the network. Each packet is divided into three different types of flits head, body and tail as shown in figure 3.4. Head flit consists of all the necessary control bits which are required of the establishment of connection between the source x and destination y. it consists of source address, destination address, control bits required for routing computation, packet size and virtual channel identifier [5]. Body flit consists of all the data require to transfer to the destination which follows the connection established

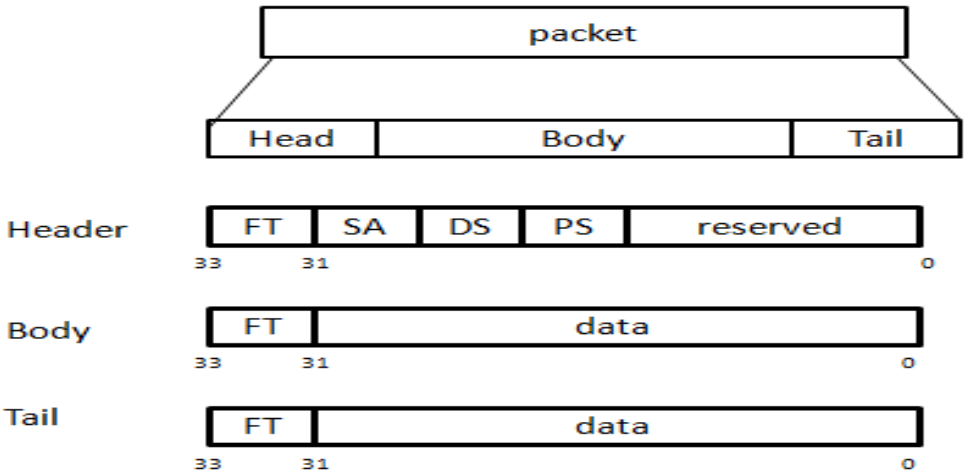

Figure 3.4: NOC packet format

SA = source address

DA = destination address

PS = packet size

by the head flit. Tail flit is required for the release off the connection and it can be utilized by other resources.

For simplicity, we have chosen only 34 bit flit with only little control information. The last two bits indicate the flit type. When the last two bits are: 2'b11 indicates head flit; 2'b10 indicates body flit; 2'b01 indicates tail flit; 2'b00 invalid flit. Out of the 34-bits of the Head flit is consisting of 8-bit source address, 8-bit destination address and 8-bit packet size. The remaining 8-bits are reserved which are used by the routers to exchange the information between the routers. For example, the reserved bits are used as virtual channel identifier in case of virtual channel router for which these 8-bits are updated from router to router as the flits traverses through the interconnect. We can add more information to the header it requires increasing the size of the flit, which in turn requires the increase of data width of buffers which are used in the network. As the wormhole switching is used all the flits of the packet fallows the same path in which the header flit traverses the interconnection to reach the destination there will be no need of sequence number for sequence ordering.

3.5. Components of NI

NI consists of two modules

1. Packing module and
2. Unpacking module.

3.5.1. Packing module

It converts the messages into flits. It assigns the flit type for each of the flit generated by packing module. The flow chart of the packing module is as shown in figure 3.5. When reset is made active it enters into IDLE state it assigns all the internal and output signals to value zero. When reset is lowered and FIFO is not empty it enters into header state and starts generating header file. After the generation of the header it checks the command (CMD) signal which is based on whether the requester wants to transfer data to a destination address

or read data from a destination address [5]. If requester wants to transfer data to destination address it enters into body state which it generates body flits based on the size of the packet, if requester wants to read data from destination then there is no need of body flit. Depending on the packet size field the body flits are generated until the packet size is reduced to one. When FIFO is empty it enters into wait state otherwise it generates the tail flit. Packing module requires 4 clock pulses to generate a 34 bit flit when input to the packing module is 8-bit.

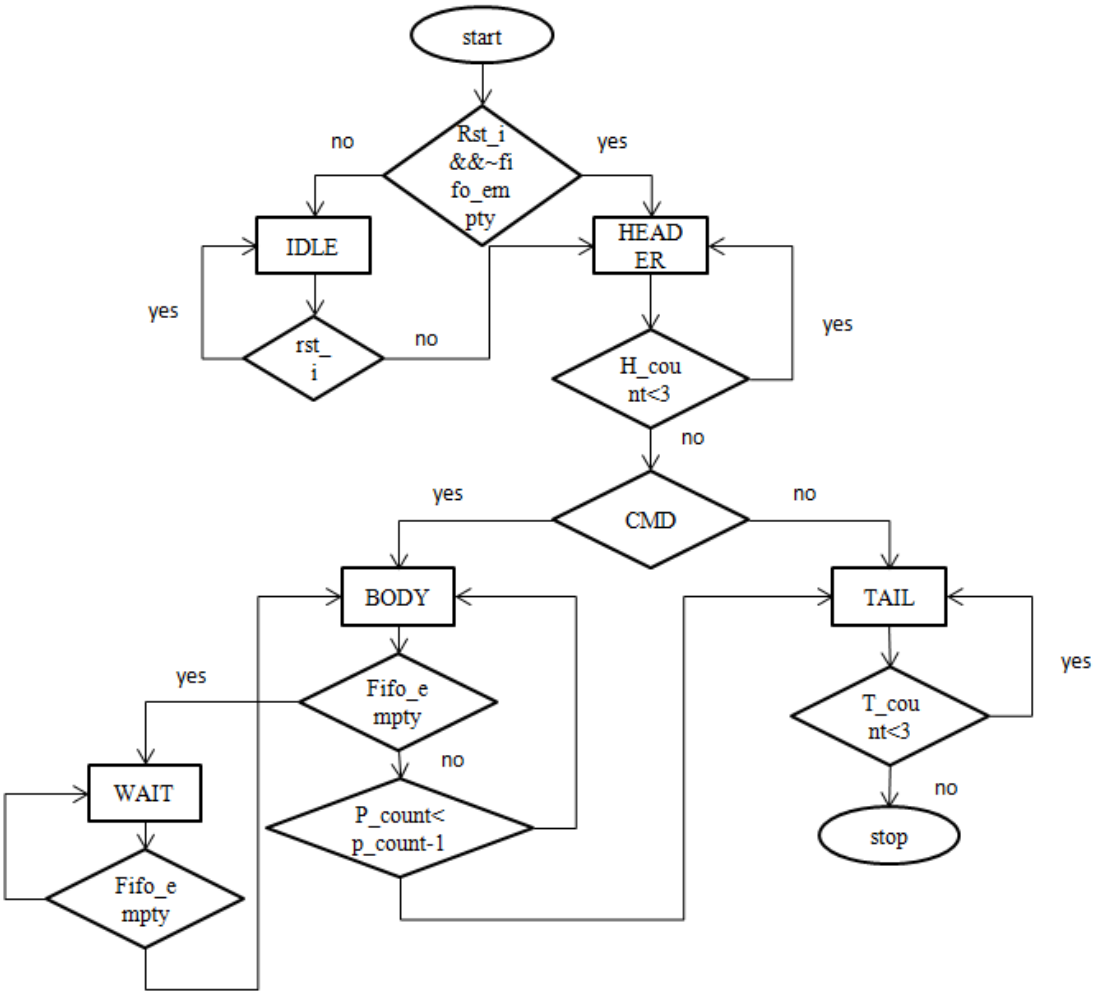

Figure 3.5: Flow chart of packing module

3.5.2. Unpacking module

It receives the flits from asynchronous FIFO. It converts the flits into messages. Based on the bits on the flit type it differentiates the header, body and tail flits. The flow chart of unpacking module is as shown in the figure 3.6. A flit that is received takes four clock cycles when input is 34 bit and output is 8 bit.

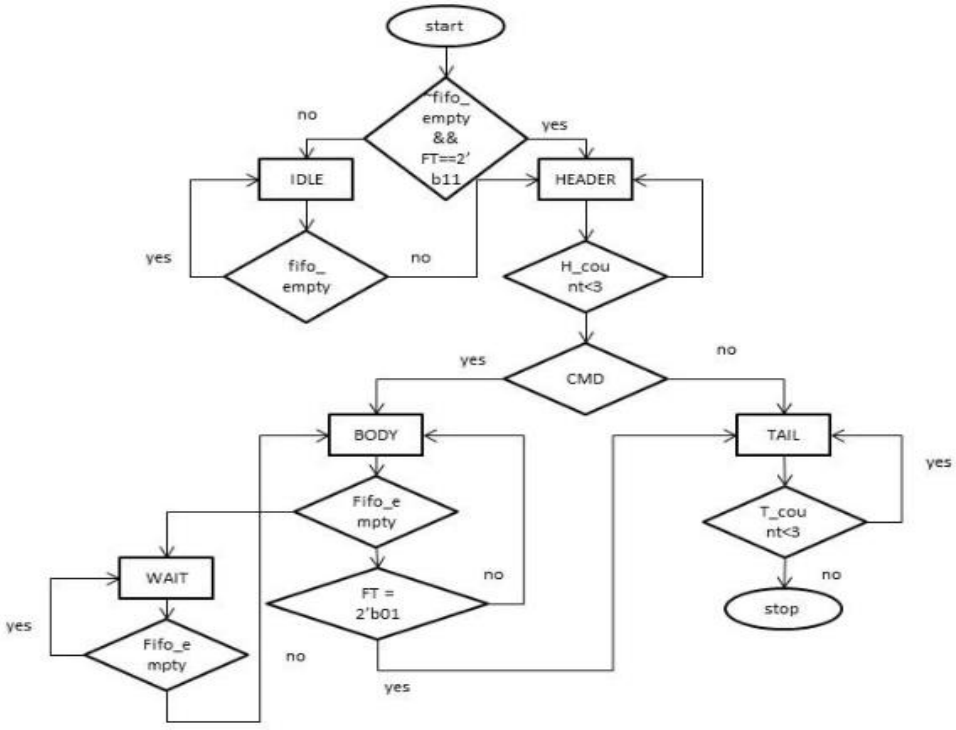

Figure 3.6: Flow chart of unpacking module.

This output is stored in a 64-bit register based on the value of sel_i signal the data width is altered and it is send to the processor. When FIFO is not empty and flit type is 2'b11 header extraction takes place. When flit type is 2'b10 it indicates body and data extraction takes place in four clock pulses when input is 34 bit and output is 8 bit. When flit type is 2'b01 it indicates tail flit which indicates the completion of the data transfer.

4. Different Router architectures

4.1. Generic router

For the worst case scenario, a general NoC router will consists of five input ports and five output ports out of which four ports are in four directions: East, West, North, and South. The remaining one port is a local port connecting to the local processor [2]. It consists of five main components to properly route the packets they are: virtual channel allocator (VCA), routing computation (RC) unit, buffers, switch allocator (SA) and crossbar network. The block diagram of the generic NoC router is as shown in figure 4.1.

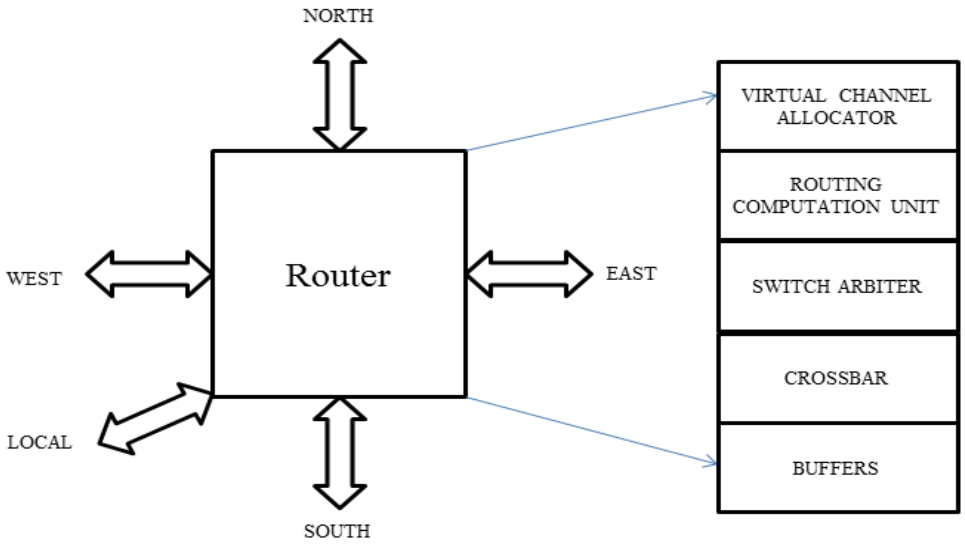

Figure 4.1: Generic NoC router

On the arrival of the header flit the RC unit starts computing the route for the flit. The VCA unit based on the control information received from the transmitting router it changes the virtual channel identifier and arbitrates among all the flits trying to access the same VC and allocates the VCs for the flits. SA arbitrates among the flits belonging to different physical channels trying to access the same resource and gives access to only one flit to traverse through the crossbar and reach the resources based on the arbitration policy. The flits belonging to different headers stored in different VCs will be passed through the same

communication physical channel through time division multiplexing manner to maintain fairness in resource allocation.

The main components of the router design are

1. Crossbar network
2. Arbiter
3. Routing computation unit

4.1.1. Crossbar network

For the design of the router a non-blocking network is required to reduce the latency. A network is said to be non-blocking, if it is able to connect all the inputs and outputs in all possible ways. To be precise, it should be able to connect any input to any output without any conflicts. Non-blocking networks are initially used for the telephone exchange. There are two types of non-blocking networks are there: if any unused input is able to connect to any unused output without changing the path that was used by any traffic is strictly non-blocking network [2]. If any unconnected input can be connected to any unconnected output, but by rerouting some unrelated traffic to make this connection is known as rearrangeble non-blocking network. Crossbar belongs to strictly non-blocking network and it is implemented as shown in the figure 4.2.

A NXM crossbar switch directly connects N inputs and M outputs without any intermediate stages. This switch consists of M N:1 multiplexers, one for each output port. Crossbar networks may be square (M=N) or rectangular if M>N or N>M. we have used a 5X5 crossbar switch for the router design as shown in above figure. The connection between an input and output port can be established by properly selecting the control signals of the multiplexers.

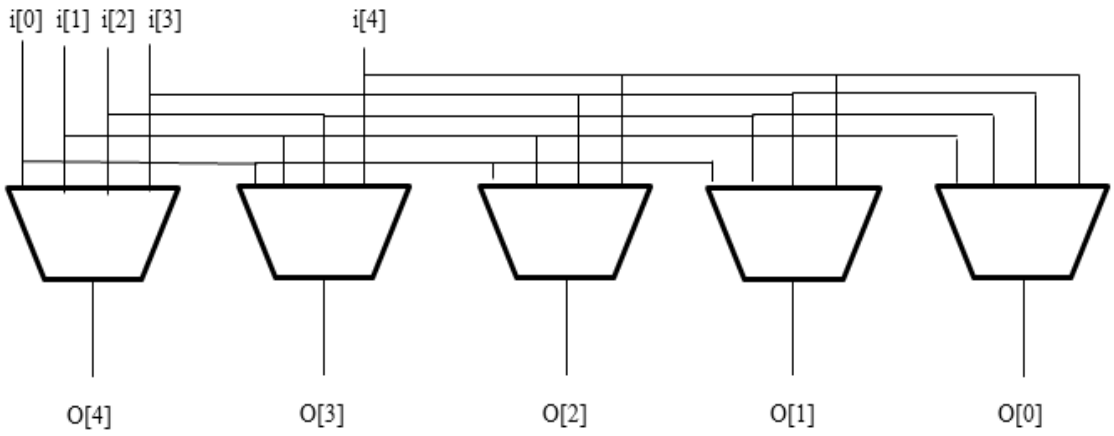

Figure 4.2: 5X5 crossbar network

4.1.2. Arbiter

Arbiter plays a vital role in the design of the router. It is used to resolve the contention problem which is caused when more than one input tries to access the same output simultaneously. Arbiter controls the arbitration of all the ports and resolves the contention problem. It keeps updated information on which ports are free and which ports are busy and then it performs the arbitration using the scheduling algorithm [8]. In the arbiter design we have used round robin scheduling algorithm. Arbitration should guarantee fairness in scheduling, avoid starvation and should allow only one block for shared resources at a time.

4.1.3. Round robin scheduling

In this scheduling algorithm the priority of an input port is varied every clock cycle. It gives access to the shared resources to the highest priority input. It consists of two stages request generation and permission granting to the requestors to the resources. This improves the fairness of arbitration. For example, if there are four inputs requesting for the same shared resources:

i. At time T_1 in[0] will have the highest priority

ii. At time T_2 in[1] will have the highest priority

iii. At time T_3 in[2] will have the highest priority

iv. At time T_4 in[3] will have the highest priority

And again at time T_5 in[0] will get the highest priority. The priority occurs in a cyclic order. If any of the highest priority input is not present then the next highest priority input will get the permission to the shared resource i.e, if at time T_1 in[0]is not requesting for the shared resource the next priority input in[1] will get permission to the resource if present otherwise for in[2] and so on.

4.2. Arbiter design for router

For achieving fairness in resource allocation we have selected round robin scheduling algorithm. It is implemented by using priority encoders and a cyclic shift register.

4.2.1. Priority encoder

It assigns priority to each of the inputs and produces the outputs accordingly. As priority is assigned to each input when more than one input arrives at a same time only the output corresponding to the highest priority is generated.

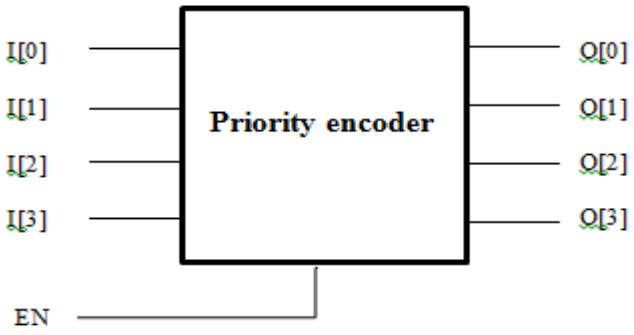

Figure 4.3: input and output signals for priority encoder

The characteristic table of the parity encoder is as below

Table 4-1 characteristic of priority encoder

EN	I[0]	I[1]	I[2]	I[3]	O[0]	O[1]	O[2]	O[3]
0	X	X	X	X	0	0	0	0
1	1	X	X	X	1	0	0	0
1	0	1	X	X	0	1	0	0
1	0	0	1	X	0	0	1	0
1	0	0	0	1	0	0	0	1

The characteristic equation of the outputs is

O[0] = EN * i[0]

O[1] = EN * (~I[0]) * i[1]

O[2] = EN * (~I[0]) * (~I[1]) * I[2]

O[3] = EN * (~I[0]) * (~I[1]) * (~I[2]) * i[3]

4.2.2. Bus arbiter design

We have designed a 4X4 bus arbiter for which it will have 4 request and 4 grant signals as shown in figure 4.4 to change the priority of the input requests for each time slot 4-bit cyclic right shift register initially loaded with 4'b1000 is used to generate the EN signal for each priority encoder.

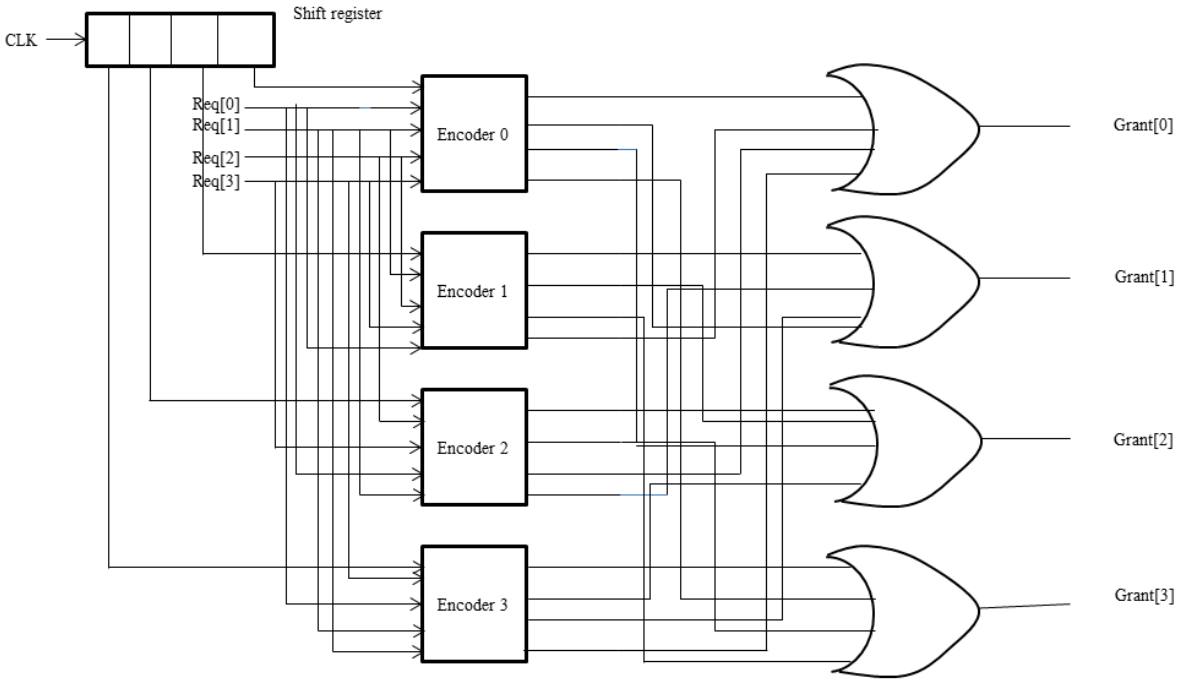

Figure 4.4: Bus arbiter structure

At a time only one priority encoder is activated and hence the priority of the input requests can be changed for each clock pulse which increases fairness and reduces starvation (an input waiting for long time as the output is kept busy by the highest priority inputs). Out of the four requesting input signals only one request is granted for each clock pulse [8]. At the output of the encoder an OR gate is used for combining all the same requests. This bus arbiter can be used in a system consisting of 4 masters and a single bus. For each clock pulse only one master is granted permission to access the bus in a round-robin manner.

4.2.3. Switch arbiter design

Bus arbiter is the basic block in design of bus of switch arbiter. We have designed a switch arbiter to accomplish arbitration in a router consisting of 5 input ports and 5 output ports which uses a 5X5 crossbar switch to avoid internal blocking. For each output port an arbiter of size 4X4 is used; a total of five 4X4 arbiters is used to perform the arbitration of the entire switch. Base on the output of the each arbiter the select lines of the crossbar is selected to grant permission for the requesting input.

The switching technique we have used is wormhole switching. In this once a path was established it should not be changed until the whole packet traverses the switch. Hence instead of changing the priority for each clock pulse, the priority is changed for each packet which is divided into several flits. The header flits arriving at the input port generates request to an output port based on the arbiter outputs the requesters are granted permission by establishing a connection in the crossbar. This connection is not changed until the tail flit traverses the switch to ensure switching is done at packet level.

4.3. Bufferless router

In a bufferless router all the input packets are sent out without buffering. The area requirement and power consumption of bufferless router is very much less. The drawback of this router is in terms of performance.

When more than one input port tries to send packets to the same output port at the same time one input port is granted permission by arbitration and the remaining packets are either dropped or deflected.

In dropping flow control mechanism when an input port does not get access to the output port the packets coming to that particular input port are dropped and they are requested for the retransmission [12]. This increases the traffic when the number of retransmissions increases and the performance of the NoC degrades which is not desired.

In deflection routing it is also called hot-potato routing instead of dropping the packets in case of contention the packets are misrouted to the other port using a deflection policy. This misrouting is referred to as deflection. Deflection policy deals with the policies to resolve contention. Contention is resolved by forwarding the highest priority packet to the favored port and misrouting the lowest priority packet to the unfavored route. This increases

latency and energy consumption due to rerouting. So, routers with buffers came into existence to store the packets when contention occurs.

4.4. Buffered router

As long as only one packet arrives at a time to an output port there will be no need of any storage elements, and the packets are routed with minimum latency. When contention occurs only one packet can be forwarded to an output port at a time, and the remaining packets are stored for later transmission. The maximum throughput at which a switch operates directly depends on how effectively the conflicting packets are stored and forwarded when the appropriate output port is no longer busy.

The buffers must be able to accept the incoming packets (write operation) and read the outgoing packets simultaneously i.e., the write and read operation of the buffer must take place simultaneously [14]. The place of buffer space can be located in three parts, buffering can be done by centralized buffers or by independent buffers at each output port or by independent buffers at each input port.

4.4.1. Buffering using centralized buffers

All the input and output ports can access the shared centralized buffer. Central buffer pools have drawbacks in terms of performance and implementation. For example, if the number of input ports is N and output ports are N, then the central buffer would require a minimum of 2N ports capable of handling 2N operations at a time. Performance is less because due to complete sharing a single congested output port may share most of the available storage in buffer pool thus preventing all other communication through the switch. Implementation is difficult since to achieve high performance the bandwidth of the

interconnection between the centralized buffer and ports should be the sum of bandwidths of all ports.

.

4.4.2. Buffering at output ports

When contention occurs, the buffers at output port must support multiple write operations. Hence, the problem with this is to accommodate the worst case situation the output port buffers should have either as many write pointers as there are input ports or the crossbar switch must operate with a speedup of sum of the input ports. Speedup is nothing but the ratio of output drain rate to the input fill rate or it can be viewed as number of input and output ports in a crossbar relative to the number of input and output ports of the router. The speedup of the network can be increased by increasing the number of crossbar connections ex: if the numbers of crossbar connections are increased by 2 then the speedup of the network will be 2. Buffering at output ports avoids head of line (HOL) blocking (explained in next section) which is caused by buffering at input ports.

4.4.3. Buffering at input ports

The advantage of input buffering is as only one packet will be arrived at a time to an input port only single write operation to the buffer at a time will be sufficient. The buffer should be capable of storing variable length packets. By using FIFO (first-in-first-out) as buffer we can achieve this.

The disadvantage of using FIFO as buffer at input ports is head of line (HOL) blocking. HOL is a phenomenon in which when one packet stored in the FIFO is blocked due to busy output port the remaining packets which are stored next to that packet will be blocked even if the output ports to which the packet is to be sent is idle. This decreases the throughput of the network.

4.5. Router design with input buffer

For an NoC the preferred packet switching method is wormhole switching. The mechanism of wormhole switching is presented in previous chapter. The advantage of this is the amount of buffers required at each input port is less and the entire packet is no need to be stored. Buffer allocation and flow control are performed at flit level in wormhole switching.

The block diagram of wormhole router is as shown in the figure 4.5. Upon the arrival of the header flit the routing computation logic extracts the header information and computes the route to the flits based on routing algorithm. We have used a XY routing algorithm.

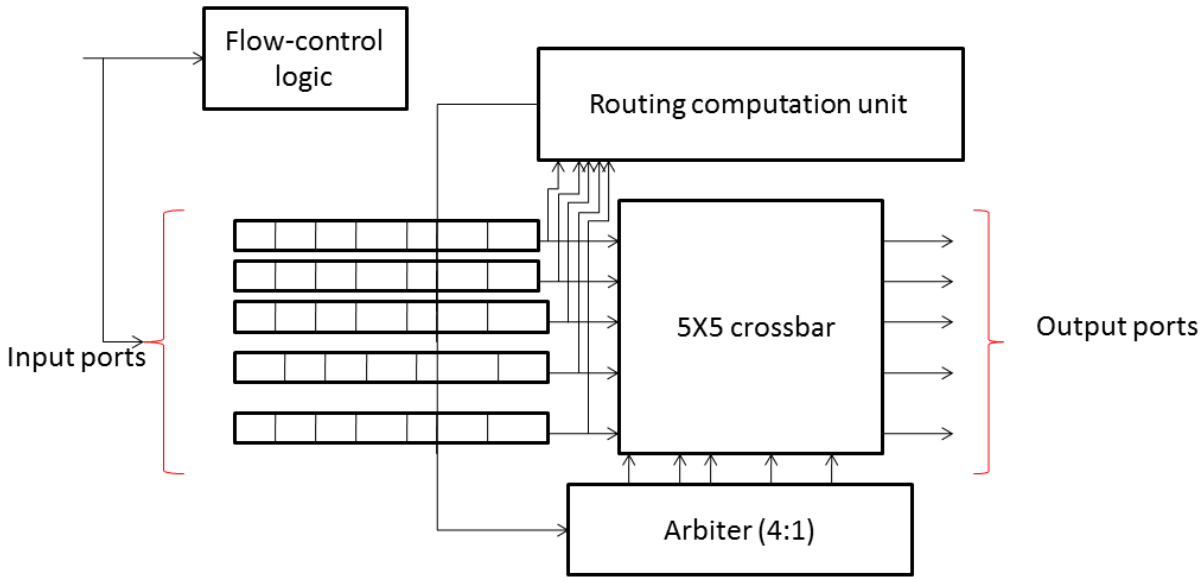

Figure 4.5: Wormhole router

The router we designed consists of five input ports and five output ports [23]. Each output port is assigned to an arbiter of size (4X1) i.e., out of four input requests only one input is granted permission based on the round robin scheduling algorithm. The outputs of the arbiter are used as select lines to the mux which are used in crossbar switch. From the values of the select lines the crossbar switch is switched and grants permission to the input requests.

The disadvantages of wormhole router are it is unable to avoid HOL and it causes deadlock.

4.5.1. Deadlock

It causes severe problems in the network avoiding the further movement of the packets. Deadlock occurs due to the cyclic dependency of the network resources which consists of buffers, channels and crossbar switches. A packet in the upstream node is moved into the downstream node only when the buffer space to store that packet is available. If the buffer space is not available that packet is stored in the upstream node and waits until the buffer space at the downstream node is available to store the packet. Similarly, the downstream node also waits to forward the packet to its downstream node if buffer space is not available at that node [13]. If this waiting continuous in a cyclic fashion then there will be no further movement of the packets and all the packets will be blocked causing deadlock. The deadlock caused in a 8X8 mesh topology is as shown in the figure 4.6.

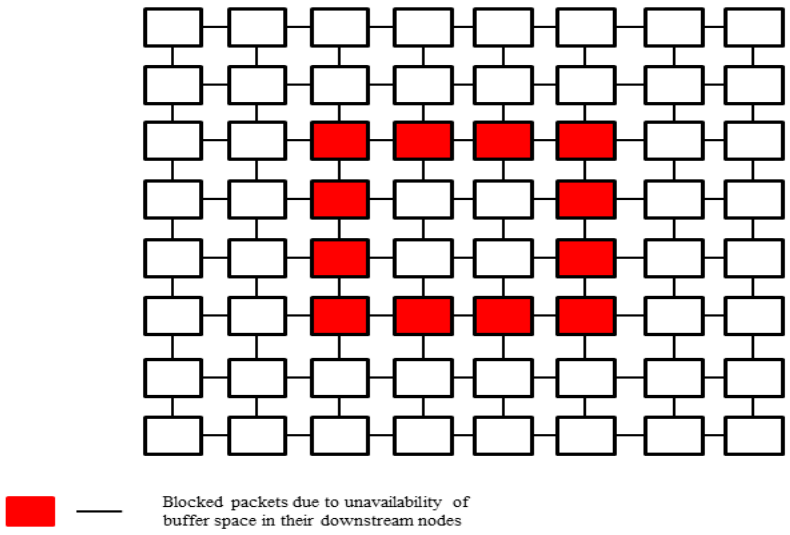

Figure 4.6: deadlock in 8X8 mesh topology

For simplicity we can treat deadlock as follow: assume that three persons are standing in three locations which are in cyclic order. For moving a person in one location to another

location, that location must be empty. As the locations are arranged in cyclic order and all the locations are filled the movement of the persons into other locations is blocked causing deadlock. Thus, the cyclic dependency of resources (locations in above case) is the cause of deadlock which will be responsible for network failure. To avoid deadlock this cyclic dependency of resources should be breakdown [13]. Many deadlock free algorithms and different router architectures are proposed to avoid deadlock. Virtual channel flow control is one among them which can effectively avoid deadlock.

4.6. Virtual channel router with full crossbar

The idling of the output ports even though there are packets directed to the idle output ports caused due to the utilization of the FIFOs at the input ports is the major drawback and reduces the throughput of the network which is undesired. To avoid the idling of the output ports and to increase the switches efficiency and throughput packets must be stored in the buffers according to the outputs they have to be routed at each input port. To implement this it was proposed to use separate FIFOs for each output port at each input port [14].

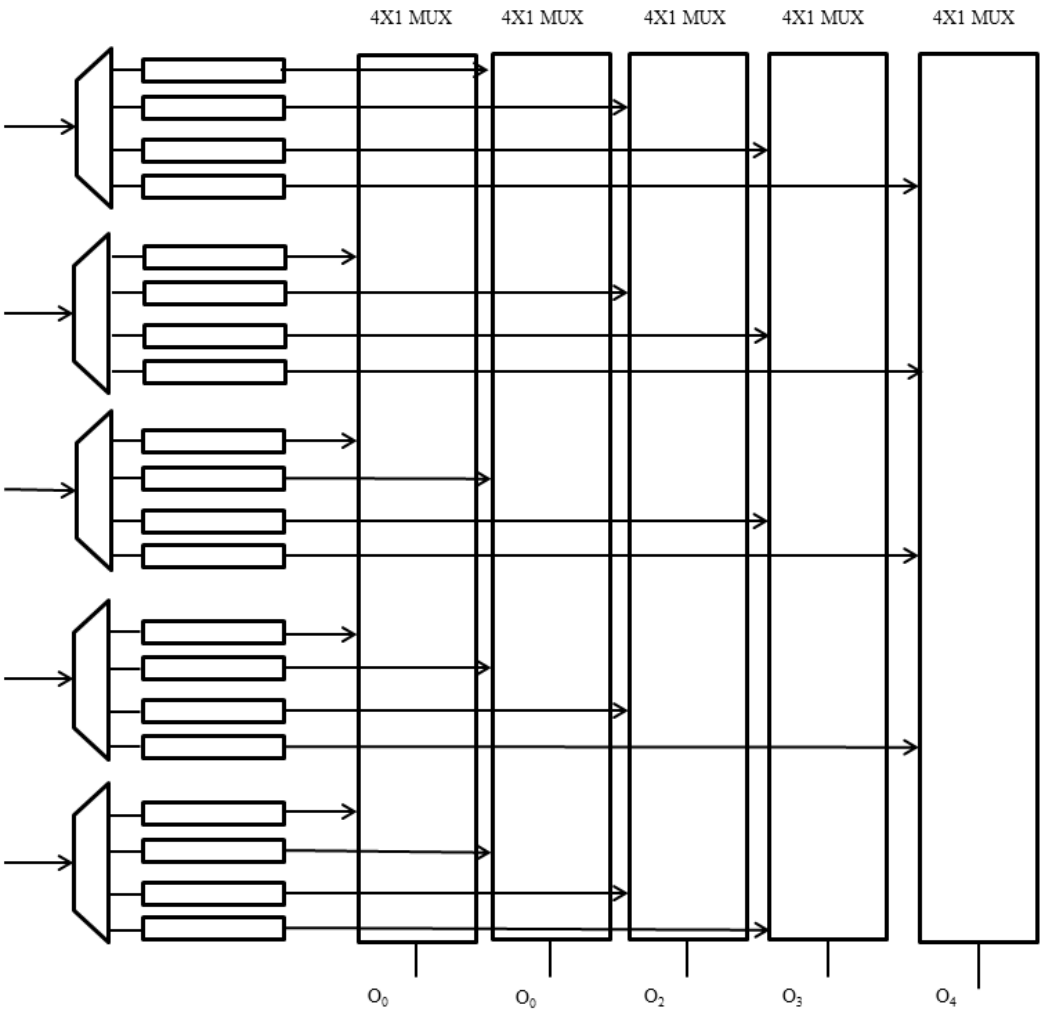

Figure 4.7: VC with full crossbar

In our designed buffer we had used 5 input ports and five output ports and the packet in an input port does not traverse the corresponding output port that is, the packets at port 1 does not traverse through the output port 1. Thus a packet at an input can be routed through one of the four output ports except to the port it has arrived. Hence it requires four buffers at each input port. The block diagram VC router with full cross bar is as shown in figure 4.7 it requires a total of twenty buffers. As there are multiple buffers at each output port a simple 5X5 crossbar switch will not be able to accommodate all the possible ways of transmission. To have all possible ways of transmission this scheme requires 20X5 crossbar network as

show in figure 4.7 five 4X1 MUXs are used. HoL blocking problem does not occur in VC router with full crossbar and it increases the throughput.

The problems with this design are it requires a large amount of overhead. Five separate crossbar networks are to be controlled instead of one and it is easy to control one 4X1 switch instead of one 5X5 switch. Using five crossbar networks it requires replicating of the same hardware five times. Buffers consume huge amount of power, this design uses four buffers and buffer controllers for each input port. The available storage cell utilization is not as good as wormhole switch due to statistically partitioning of the buffer space

Implementing flow control for this design is another difficult task. The upstream router must be notified for not transmitting the packets until the buffer is free when the downstream buffer is full. As each input port consists of four buffers, four buffer control signals must be transmitted between the upstream routers which are four times the control information passed by the wormhole router. Most of the channel bandwidth is consumed for the transfer of control signals. When a buffer is full the upstream router can pre-route the packet to a buffer which is not full, but increases the hardware complexity of the router. In this design the router waits until the buffer frees up to accept the packet.

4.7. VC router

It assigns multiple virtual paths to the same physical channel. The bandwidth of a single physical channel can be used by multiple virtual channels. By providing multiple buffers for each channel, virtual channels are able to decouple the allocation of buffers from channels allocation. For a single channel there will be multiple buffers providing multiple virtual channels [11]. The allocation of the virtual channel to a flit depends on the virtual channel identifier (VCI). VC router gives 100 percent throughput by avoiding the deadlock and HoL. The block diagram of the virtual channel router is as shown in figure 4.8.

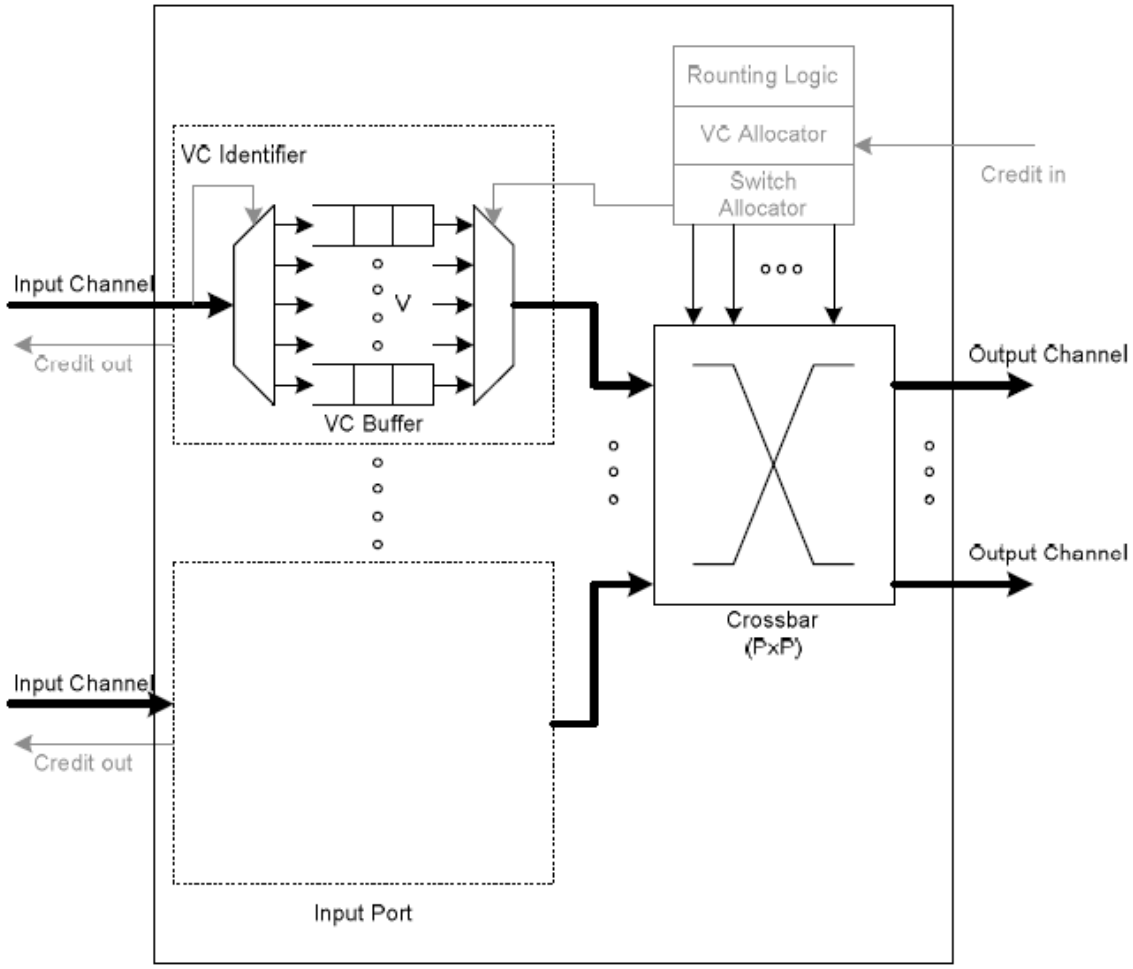

Figure 4.8: virtual channel router

Many factors are to be considered while using virtual channel router as a transmitting router. Each router has to exchange the credit information which includes how many VCs are available in neighbouring routers and the numbers of free buffers are available. The receiving router should keep track on the availability of buffer space and also should update the read and write pointers. VCA is used to assign a virtual channel to the incoming flits and switch allocator is used to assign an output to the flits stored in the virtual channel buffers.

4.7.1. VC allocator design

The complexity in designing the virtual channel allocator depends on the number of virtual channels returned by the routing algorithm [17].

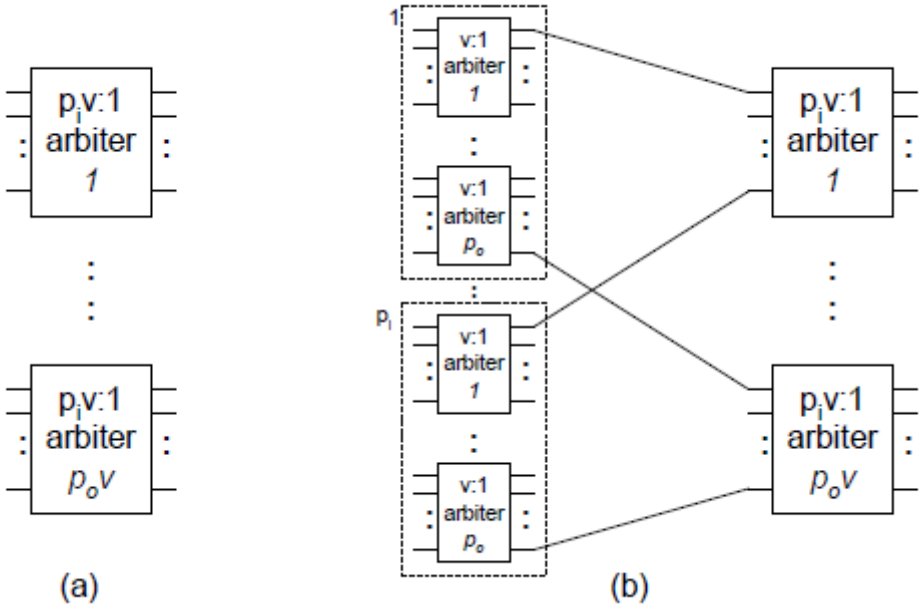

Figure 4.9: VCA for routing function which returns a single VC. b) VCA for routing function which returns virtual channels of a single physical channel.

In the above figures P_i represents the number of input ports, P_o represents the number of output ports, V represents number of virtual channels figure 4.9 a) shows the VCA for he routing function (R→v) which returns a single virtual channel [15]. In this case the VCA needs to arbitrate among the input virtual channels which are requesting for the same output virtual channel. If the routing function returns any of the virtual channels of a single physical communication channel (R⟶p), then the VCA is as shown in the above figure 4.9 (b) in the first stage of arbitration each of the arbiters should arbitrate among v possible requests, before forwarding to the second stage arbiters.

4.7.2. Switch arbiter design for VC router

The task of the switch arbiter is to arbitrate among all VCs requesting the outputs and to grant permission to the requests in a manner it will resolve contention. The block diagram of switch arbiter is as shown in the figure 4.10.

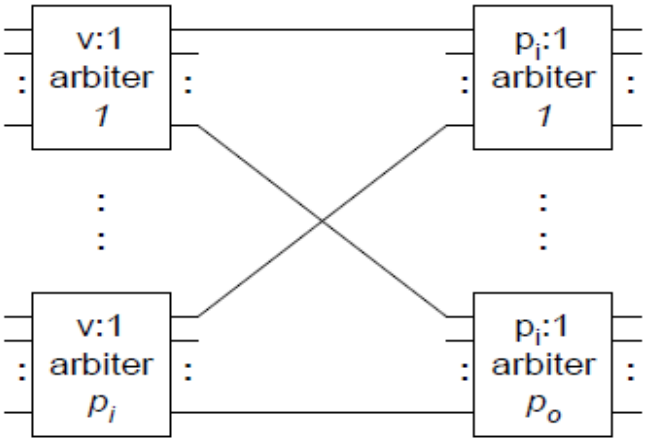

Figure 4.10: Switch arbiter for VC router

For each input port there will be an V:1 arbiter which arbitrates among all the v VCs of a port and grants permission to one VC for an input port [15]. The second stage of arbiters contains a P_i:1 arbiter for each output port and arbitrates among all the requests coming from the first stage of arbiters and grants permission in a manner to avoid contention.

5. Simulation results

We have designed all the different types of routers mentioned in the previous chapters using Verilog HDL language in Xilinx ISE tools and for synthesizing the design the FPGA family selected is SPARTAN-3e with device being XC3s500e having the package FG320.

5.1. FIFO module

The design of this module is discussed in section 3.2 the simulation results and device utilization table is presented in this section.

The designed FIFO has a data width of 10 bit and depth of 16 bits. When clear_in is set FIFO pointers are set to the initial location and fifo_empty will be high. When clear_in is low and wr_en is set the write pointer starts increasing and the writing operation takes place at a speed of write clock clock frequency and fifo_empty will be low. When rd_en is made high and fifo_empty is low read operation takes place with the speed of read clock frequency. The simulate waveform is as shown in figure 5.1.

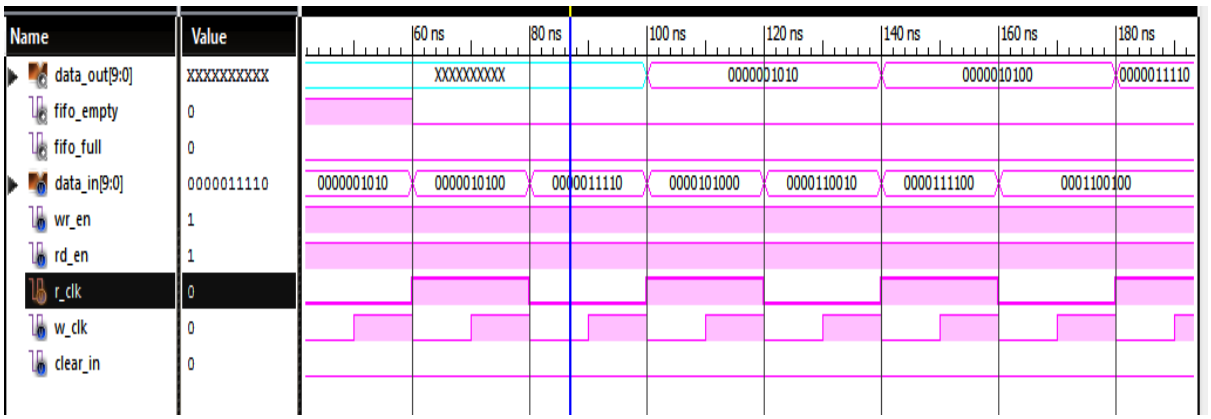

Figure 5.1: Simulated waveform of FIFO

When FIFO is full fifo_full will be set to high otherwise it is set to low. The data_out shown in above figure is the data which is reading out of the FIFO. When the FIFO is not empty simultaneous read and write operations are possible. Once FIFO is full and still data is coming to the FIFO then the data will be overwritten.

Table 5-1device utilization summary of FIFO

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	29	4656	0%
Number of Slice Flip Flops	27	9312	0%
Number of 4 input LUTs	41	9312	0%
Number of bonded IOBs	27	232	11%
Number of GCLKs	2	24	8%

5.2. NI module

The design of the NI module was explained in section 3.5 the simulated output waveform and design utilization summary table are presented here.

5.2.1. Packing module

Figure 5.2 shows the simulated wave form of packing module. Here input is 8-bit and the generated flit size is 34-bit. At the beginning of new flit the output signal valid is made high to indicate the validation of data. From the waveform we can see that it requires 4 clock pulses to generate a 34-bit flit when the input flit size is 8-bit. NI module adds two extra bits to indicate the type of flit.

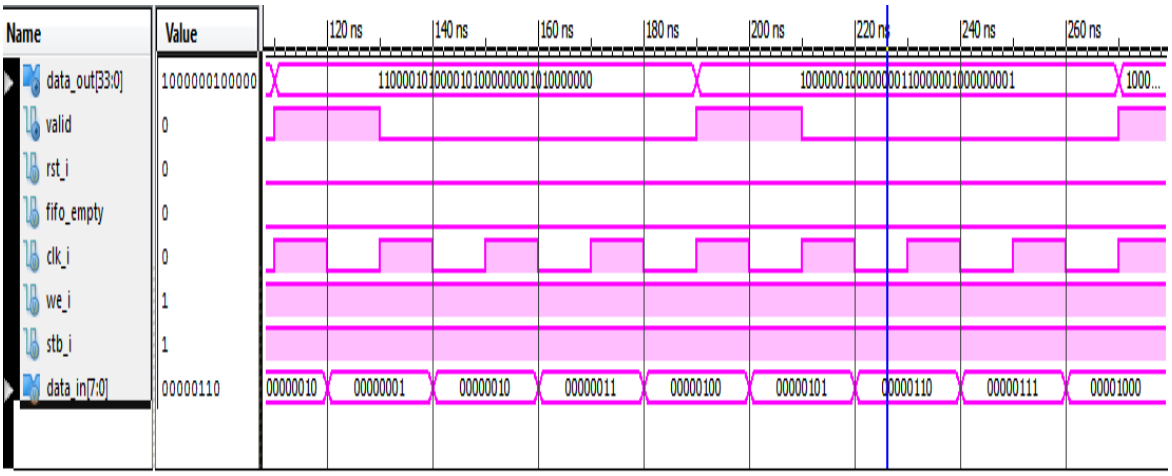

Figure 5.2: output waveform of packing module

The device utilization summary is as shown in the table 5.2.

Table 5-2 device utilization summary for packet maker

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	111	4656	2%
Number of Slice Flip Flops	101	9312	1%
Number of 4 input LUTs	206	9312	2%
Number of bonded IOBs	48	232	20%
Number of GCLKs	1	24	4%

Figure 5.3 shows the simulated wave form for unpacking module. For this input is a flit of size 34 bit and generates an output of 8-bit. On rising edge of valid signal it latches the incoming flit and it decodes into 8-bit data it takes four clock pulses. A 64 bit register is used to feed the data to PE based on the sel_i signal. The device utilization summary is as shown in the table

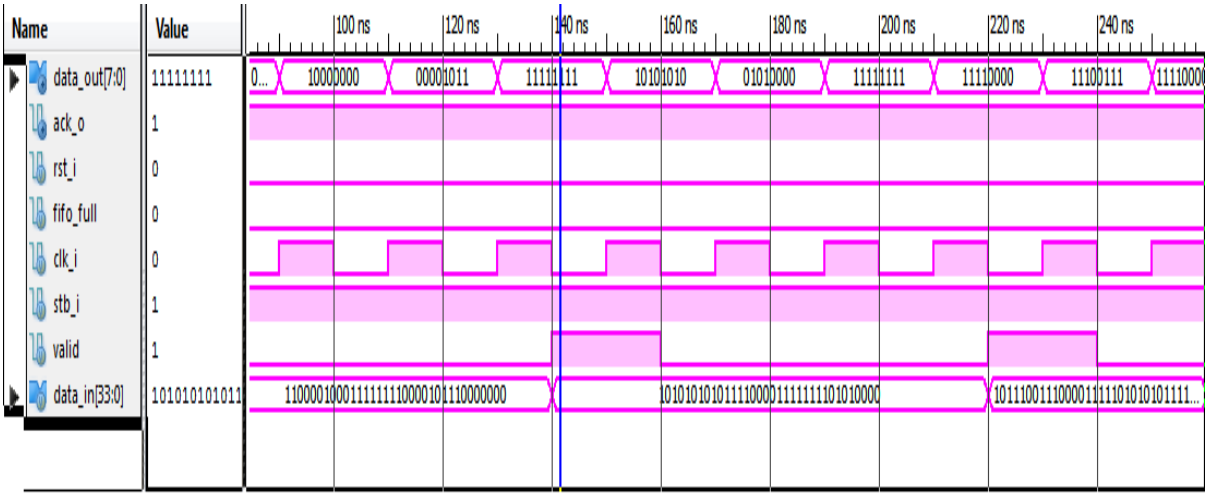

Figure 5.3: output waveform for unpacking module

Table 5-3 device utilization summary for unpacking module

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	29	4656	0%
Number of Slice Flip Flops	10	9312	0%
Number of 4 input LUTs	20	9312	0%
Number of bonded IOBs	46	232	19%
Number of GCLKs	2	24	8%

5.3. Arbiter module

The design and use of the arbiter module is discussed in section 4.2 the simulation results and device utilization summary is shown in this section.

For the understanding the working of the arbiter we have used a 8-bit data input request signal and 8-bit data output grant signal. In the design of the router we used 1-bit request input signal and 1-bit grant output signal.

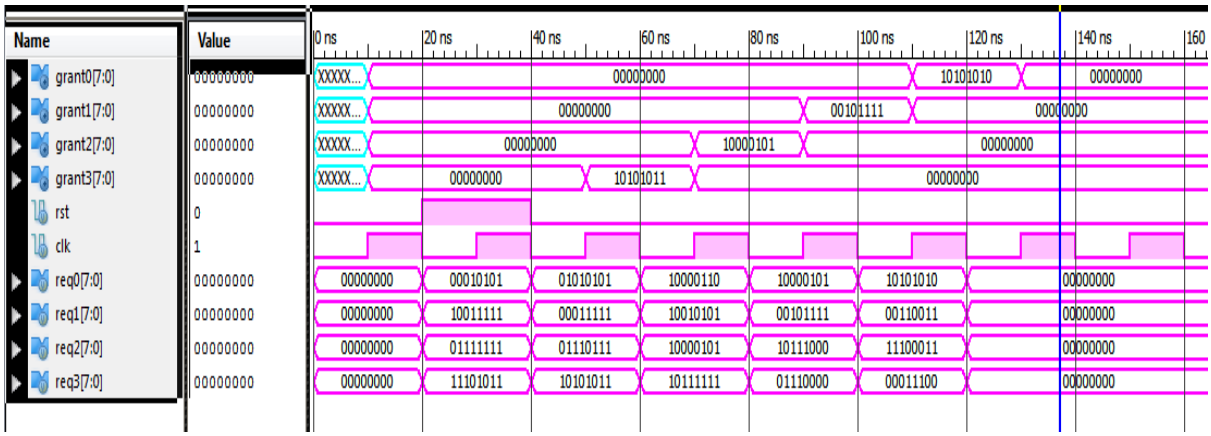

Figure 5.4: output waveform of arbiter

When the reset is high it ignores all the requests and does not grant permission to any of the requests as shown in the figure when the reset is low the requests are granted in a round robin manner in a cycle by cycle fashion. In the design of router as wormhole

switching is used this cycle by cycle shifting of priority is not used it requires to change the priority in packet by packet manner. Since, once the connection is established for the header flit to traverse the switch all the remaining flits should follow the same path. On the encounter of the tail flit the connection is realised. In the router design header sends request signal to the arbiter. Once the request is granted the same path is established until tail flit sends a connection release request to the arbiter. Upon encounter of this signal priority is changed to other port and the same procedure is repeated. The FPGA device utilization summary is as shown in the table 5.4.

Table 5-4 device utilization of the arbiter

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	22	4656	0%
Number of Slice Flip Flops	36	9312	0%
Number of 4 input LUTs	13	9312	0%
Number of bonded IOBs	66	232	28%
Number of GCLKs	1	24	4%

At the rising edge of the clock the first priority is given to the fourth port. As in figure 5.4 even though all the input ports are requesting the permission for the outputs only one input request is granted permission. At the next clock edge third port gets access to output port if request is present else port 2 will get access to the requesting output port. This process will occur in a cyclic manner. In the design of the router this kind of 1-bit arbiter is used for each of the output port, requiring a total of five arbiters forming a switch arbiter. These arbiters will generate select lines for the crossbar switch for establishing connection.

5.4. Bufferless router module

The design of the bufferless router is explained in section 4.3 the device utilization summary and simulated waveforms are presented in this section.

The input to the bufferless router is a 10-bit data and output is a 10-bit data. The last two bits of the flit indicate the flit type. When reset is high all output ports are idle. The header flit will be the first flit to arrive at the input port it contains the destination port address. Based on the destination address the crossbar network is established and the packets are routed to the requesting output ports if the output port is idle.

Table 5-5 router port addresses

S. No	PORT NAME	PORT ADDRESS
0	LOCAL	0011
1	EAST	0001
2	WEST	0010
3	NORTH	0100
4	SOUTH	1000

The packets are dropped when output port is busy. This can be seen in the figure 5.5, at beginning all the header flits of the packets are directed to different ports and all the ports are busy there is no dropping and all the ports are utilized effectively. At the arrival of second header flit at the input ports at the same instance dropping of packets occurred due to input port0 and input port 1 are trying to access the same output port 2 and the input ports 2 and 3 trying to access the same output port 0.

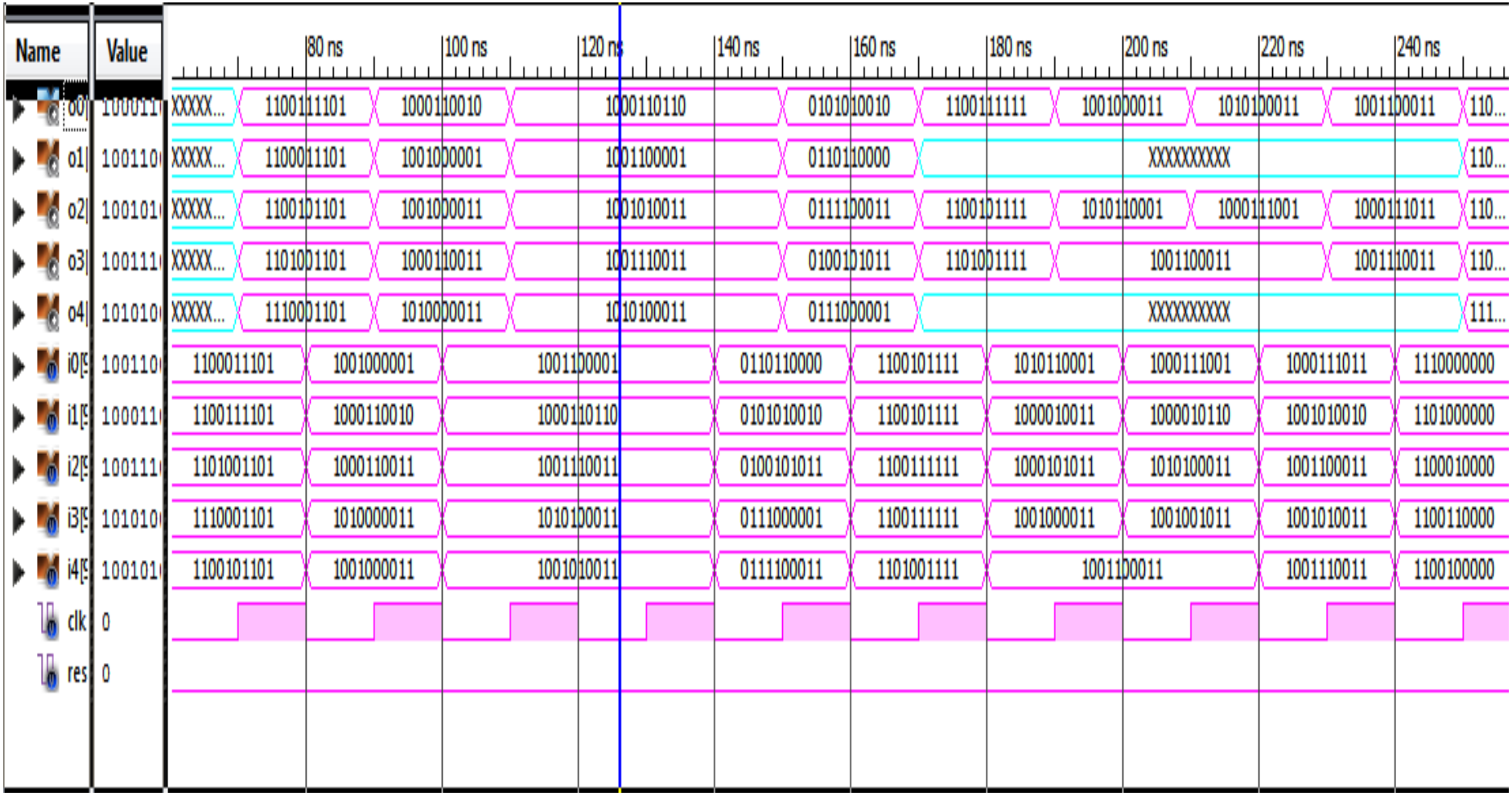

Figure 5.5: output waveform of bufferless router

The device utilization summary for the bufferless router is as shown in the table 5.6. From this it is clear that bufferless router consumes only little amount of the area.

Table 5-6 device utilization summary for bufferless router

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	3	4656	0%
Number of 4 input LUTs	6	9312	0%
Number of bonded IOBs	11	232	4%

The design of the buffered router is presented in section 4.5 the simulated waveforms and the device utilization summary of the buffered router is shown in this section.

This router routes the 10-bit size incoming flits. The flits are buffered temporarily before they are routed. When there is contention the packets are stored in the buffers and are transferred when the output port becomes idle. This can be seen in the figure 5.6 the header flits of input port 0 and por1 are trying to access the same output port 2. One port is got accessed and the other is stored in the buffer temporarily. When packet at input port completes its transmission it gives the access to the store packet at input port 1. The drawback of wormhole router which is HoL can also be seen in this figure. The input port 1 is containing the flits which are destined to the output port 3. Even though the output port 3 is idle it cannot get access to that port due to the blocked flits which arrived earlier than this flit which is trying to access the output port 3. Due to this the resources are underutilized and the throughput of the router is also decreases. The device utilization summary of the buffered router is as shown in the table 5.7, it shows that buffered router requires large amount of FPGA resources required than the bufferless router.

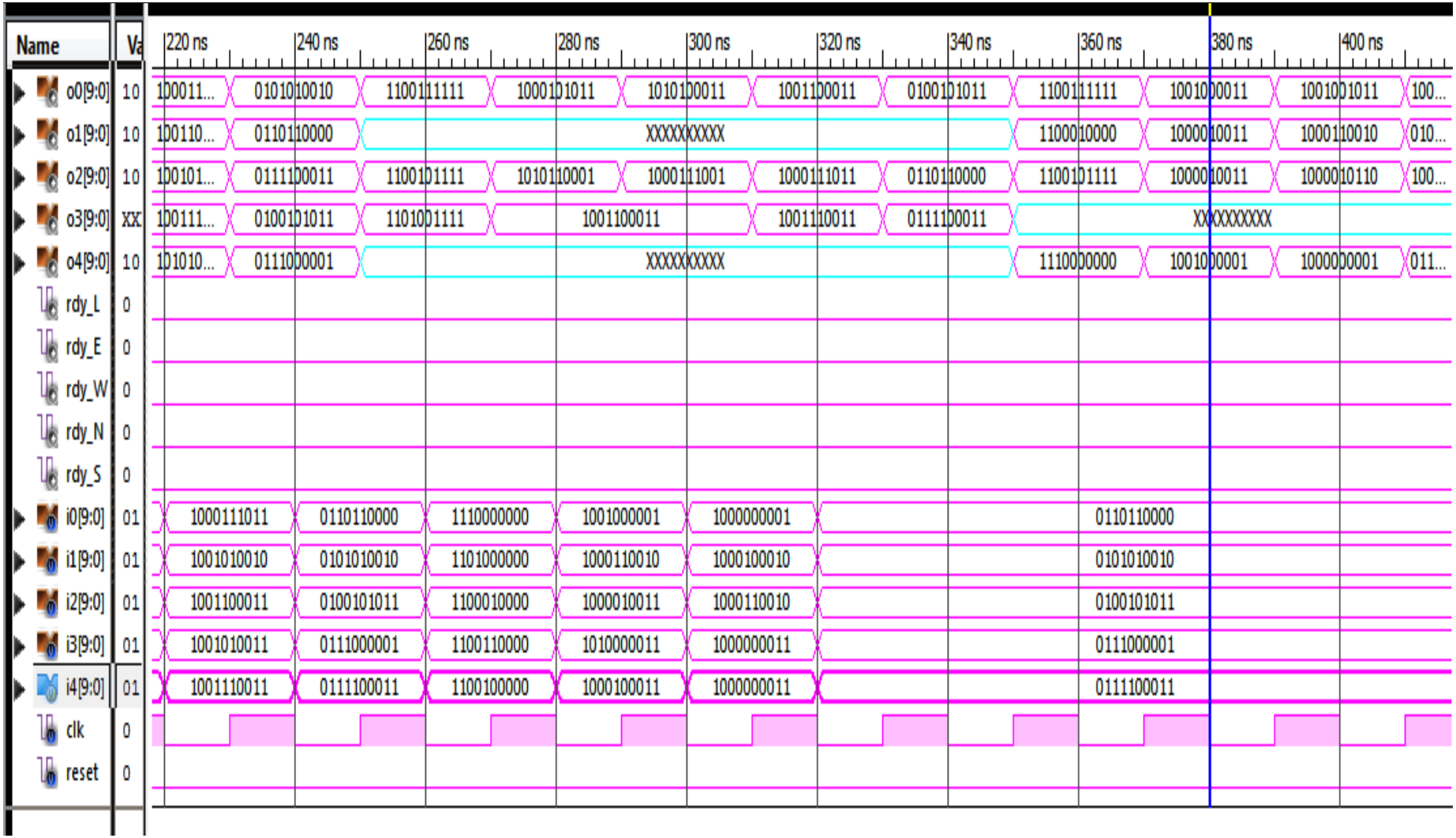

Figure 5.6: output waveform of wormhole router

Table 5-7 device utilization summary of the wormhole router

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	348	4656	7%
Number of Slice Flip Flops	325	9312	3%
Number of 4 input LUTs	609	9312	6%
Number of bonded IOBs	107	232	46%
Number of GCLKs	1	24	4%

5.5. VC router with full crossbar

The design of the VC router with full crossbar is explained in section 4.6 the device utilization summary for VC router with full crossbar and the simulated waveform is shown in this section.

For all the different routers designed same inputs are used for the simulation of the waveform. The simulated output waveform for the wormhole router and the fully connected crossbar router seems to be same except in the case of HoL blocking. The chosen inputs are such that the second header flits arriving at the input ports causes contention and are stored in buffers. The next header is chosen such that an output port is made idle and for that output port there will be packets stored previously in the buffers. As the flits finds the output port to be idle the flits which are stored previously are routed to that output port without blocking, thus avoiding HoL blocking. This is shown in the figure 5.7 in this wave form the previous stored inputs are not visible, as it is not possible to show in a single waveform. Only the outputs which shows the output ports are busy even if there are no inputs corresponding to that outputs and the blocked packets does not blocking the others packets which can traverse through are shown.

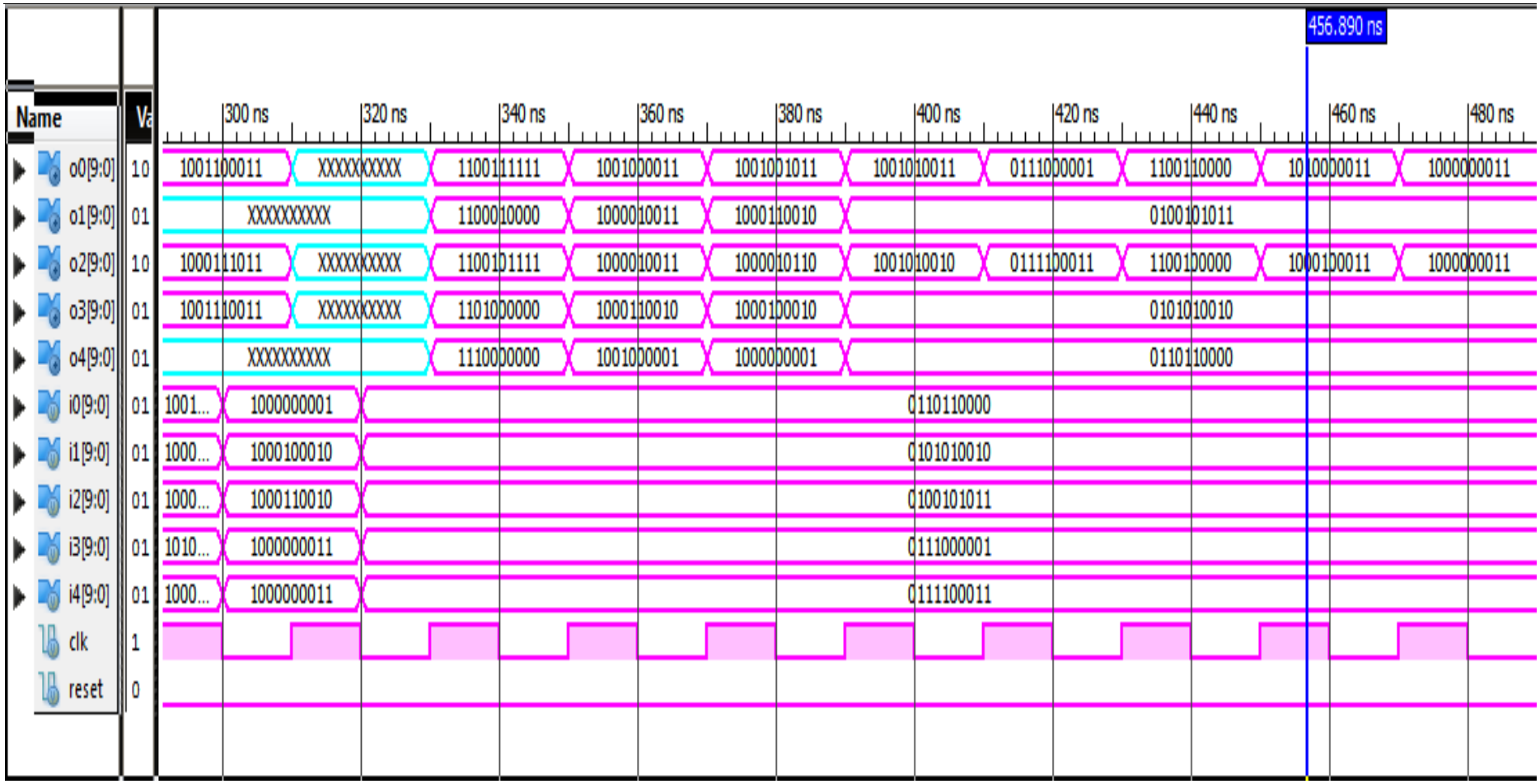

Figure 5.7: simulated waveform of VC router with full crossbar

The device utilization summary for this router is shown as in the table 5.8.

Table 5-8 device utilization summary for VC router with full crossbar

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	713	4656	15%
Number of Slice Flip Flops	630	9312	6%
Number of 4 input LUTs	1270	9312	13%
Number of bonded IOBs	102	232	43%
Number of GCLKs	1	24	4%

5.6. VC router

The design of the VC router is presented in the section 4.7, the device utilization summary is shown in this section.

The output waveform simulated for the VC router will be same as the simulated waveform generated for the VC router with full crossbar. Hence only the device utilization summary is shown and it is as per table 5.9. In this design four virtual channels for a single physical channel is used.

Table 5-9 Device utilization summary for VC router with 4 VCs

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	867	4656	18%
Number of Slice Flip Flops	602	9312	6%
Number of 4 input LUTs	2085	9312	22%
Number of bonded IOBs	216	232	93%
Number of GCLKs	1	24	4%

5.7. Power and area estimation

The Wishbone compatible NI and all the different router designs discussed in the previous chapters are synthesized by using Synopsys Design Compiler targeted to TCBN 65nm GPLUS CMOS standard cell library with 1V voltage and 25◦C temperature.

The graphs in figure 5.8 a and b show the variation of area and power with FIFO depth. The asynchronous FIFO depth has major impact on the performance of the NI and occupies more area and power than NI.

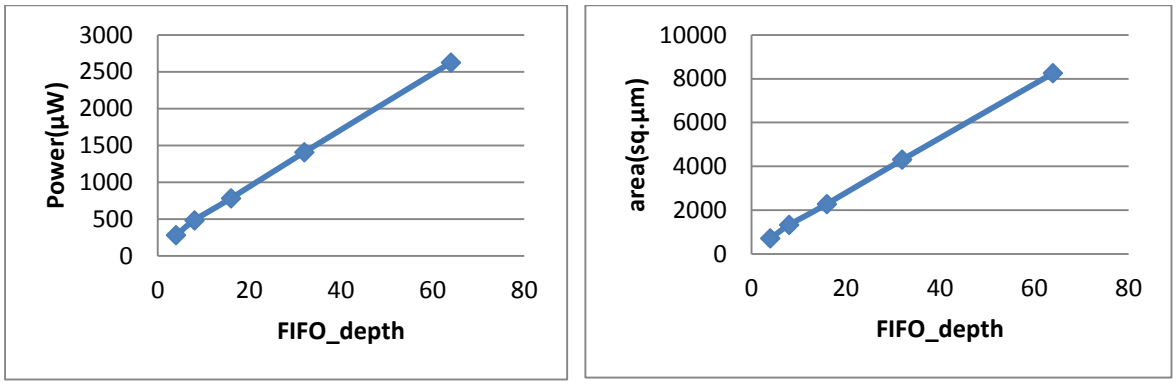

Figure 5.8: a) FIFO depth Vs power b) FIFO depth Vs area

The estimated area and power analysis of packing and unpacking module is as shown in table 5.10.

Table 5-10 Area and power of NI modules

Module name	Area (μm^2)	Total Power (µW) (static+dynamic)
Packing module	1302.47	92.8932
Unpacking module	431.64	29.704

The estimated power and are of different router architectures is as shown in the table 5.11

Table 5-11 area and power evaluation of different routers

Module	Area (μm^2)	Total Power (µW) (static+dynamic)
Bufferless router	1375.20	468.02
Wormhole router	14693.04	4309.8
VC router with 2 VCs	25326.73	7205.1
VC router with 4 VCs	48594.71	12903.6
VC router with full crossbar	47738.16	10872.32

6. Conclusion

NoC can effectively offer the communication requirements between the PEs of SoC than the conventional communication system. NoC is effectively scalable, can operate at high frequencies, requires less latency, offers reliability and flexibility. NoC design requires a large number of resources which increases cost, area and power consumption. The effective utilization of the NoC resources plays a vital role in the design of NoC which is determined by the flow control this intern reduces cost, area and power consumption.

In this project, the various components of NoC which includes asynchronous FIFO which is required for effectively transferring the data between different clock pulses and also used as buffers to store packets in a router, network interface (packing and unpacking module) which plays the role of converting data received from the PEs to flits at the source end and flits into data at the destination end, arbiter which is used to resolve contention and generates the select signals for switching, crossbar network which offers connection between the input and output ports of the router based on the outputs of the arbiter and mainly focusing on different types of NoC router including bufferless, buffered, VC router with full crossbar and VC router were designed and analysed using VERILOG HDL language with the help of Xilinx tools and verified its functionality in SPARTAN-3e FPGA kit. The area and power consumed by the designed components of NoC were estimated using SYNOPSYS design tool.

6.1. Scope for future work

The design of the router can be enhanced such that the network can be free from starvation, deadlock and HoL blocking without increasing its area and power consumption. The FIFO which are used at the input of the router to store the data and also responsible for the problems faced by the routers is to be replaced by some logic circuit such that the above problems will be rectified without any increase in area and power.

BIBLIOGRAPHY

[1] Dally W, Towles B. “Route packets not wires: on-chip interconnection networks” In: Annual design automation conference; 2001

[2] W. Dally, B. Towles “Principles and practices of interconnection networks” Morgan Kaufmann Pub, San Francisco (CA) (2004).

[3] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," IEEE *Computer*, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[4] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," IEEE *Computer*, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[5] K. Swaminathan , G. Lakshminarayanan , Seok-Bum Ko “Design and verification of an efficient WISHBONE-based network interface for network on chip” Elsevier 14 may 2014.

[6] OpenCores “WISHBONE Specification, WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores”. OpenCores. rev. B.4; 2010.

[7] Cummings C, Alfke P. “Simulation and synthesis techniques for asynchronous FIFO design with asynchronous pointer comparison”. In: SNUG; 2002

[8] Eung ***S.*** Shin, Vincent **J.** Mooney **111** and George F. Riley “Round-Robin arbiter and design generation” http://www.ece.gatech.edu/

[9] Sanjay Pratap Singh, Shilpa Bhoj, Dheera Balasubramanian, Tanvi Nagda, Dinesh Bhatia, and Poras Balsara.”Generic network interfaces for plug and play NOC based architectures” Springer-Verlag Berlin Heidelberg 2006.

[10] Khalid Latif1;3, Tiberiu Seceleanu2, Hannu Tenhunen “Power and Area Efficient Design of Network-on-Chip Router Through Utilization of Idle Buffers” 17th IEEE International Conference 2010.

[11] William J. Dally “Virtual Channel flow control” IEEE transactions on parallel and distributed systems, VOL., 3, NO. 2, march 1992.

[12] Yuhai Li, Kuizhi Mei , Yuehu Liu, Nanning Zheng , Yi Xu “LDBR: Low-deflection bufferless router for cost-sensitive network-on-chip design” ELESVIER Micro-processers and Micro-systems 38 (2014).

[13] WILLIAM J. DALLY, AND CHARLES L. SEITZ,” Deadlock-Free Message Routing in Multiprocessor Interconnection Networks” IEEE TRANSACTIONS ON COMPUTERS, VOL. C-36, NO. 5, MAY 1987

[14] Yuval Tamir and Gregory L. Frazier*”* HIGH-PERFORMANCE MULTI-QUEUE BUFFERS FOR VLSI COMMUNICATION SWITCHES**”** 15th Annual International Symposium on Computer Architecture, Honolulu, Hawaii, pp. 343-354, May 1988.

[15] Li-Shiuan Peh and William J. Dally “A Delay Model and Speculative Architecture for Pipelined Routers” In Proceedings of the 7th International Symposium on High-Performance Computer Architecture, Jan. 22-24, 2001, Monterrey, Mexico, pp. 255-266.

[16] Chrysostomos A. Nicopoulos, Dongkook Park, Jongman Kim, N. Vijaykrishnan, Mazin S. Yousif, Chita R. Das “ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers” The 39th Annual IEEE/ACM International Symposium on Microarchitecture 2006.

[17] Daniel U. Becker and William J. Dally “Allocator Implementations for Network-on-Chip Routers” ACM/IEEE Conference on High Performance Computing, Networking 2009.

[18] Bo Zhao, Youtao Zhang and Jun Yang “A Speculative Arbiter Design to Enable High-Frequency Many-VC Router in NoCs” IEEE 2013.

[19] J. Duato, J. Flich, and T. Nachiondo “A Cost-Effective Technique to Reduce HOL Blocking in Single-Stage and Multistage Switch Fabrics” Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing 2004

[20] Anh T. Tran and Bevan M. Baas “Achieving High-Performance On-Chip Networks With Shared-Buffer Routers” IEEE transactions on very large scale integration (vlsi) systems, vol. 22, no. 6, june 2014.

[21] SUNGHO PARK " A VERIOG-HDL IMPLEMENTATION OF VIRTUAL CHANNELS IN A NETWORK-ON-CHIP ROUTER” A Thesis Submitted to the Office of Graduate Studies of Texas, August 2008.

[22] Pedro J. García, Francisco J. Quiles “EFFICIENT, SCALABLE CONGESTION MANAGEMENT FOR INTERCONNECTION NETWORKS” IEEE Computer Society 2006.

[23] Zhonghai Lu “Using Wormhole Switching for Networks on Chip: Feasibility Analysis and Microarchitecture Adaptation**”** Thesis submitted to the Royal Institute of Technology 2005.

.